```
PPPPPPPPPPPP     RRRRRRRRRRRR     TTTTTTTTTTTTTTT     SSSSSSSSSSS  MMM         MMM  BBBBBBBBBBBB
PPPPPPPPPPPPP    RRRRRRRRRRRRR    TTTTTTTTTTTTTTT     SSSSSSSSSSSS MMM         MMM  BBBBBBBBBBBBB
PPPPPPPPPPPPP    RRRRRRRRRRRRR    TTTTTTTTTTTTTTT     SSSSSSSSSSSS MMM         MMM  BBBBBBBBBBBBB
PPP         PPP  RRR         RRR        TTT          SSS          MMMMMMM   MMMMMMM  BBB         BBB
PPP         PPP  RRR         RRR        TTT          SSS          MMMMMMM   MMMMMMM  BBB         BBB
PPP         PPP  RRR         RRR        TTT          SSS          MMMMMMM   MMMMMMM  BBB         BBB
PPP         PPP  RRR         RRR        TTT          SSS          MMM MMM MMM MMM  BBB         BBB
PPP         PPP  RRR         RRR        TTT          SSS          MMM MMM MMM MMM  BBB         BBB
PPP         PPP  RRR         RRR        TTT          SSS          MMM MMM MMM MMM  BBB         BBB
PPPPPPPPPPPP     RRRRRRRRRRRR            TTT           SSSSSSSSS   MMM     MMM  BBBBBBBBBBBB
PPPPPPPPPPPP     RRRRRRRRRRRRR           TTT           SSSSSSSSS   MMM     MMM  BBBBBBBBBBBBB
PPPPPPPPPPPP     RRRRRRRRRRRRR           TTT           SSSSSSSSS   MMM     MMM  BBBBBBBBBBBBB
PPP              RRR   RRR                TTT                 SSS  MMM     MMM  BBB         BBB
PPP              RRR   RRR                TTT                 SSS  MMM     MMM  BBB         BBB
PPP              RRR   RRR                TTT                 SSS  MMM     MMM  BBB         BBB
PPP              RRR      RRR             TTT                 SSS  MMM     MMM  BBB         BBB
PPP              RRR      RRR             TTT                 SSS  MMM     MMM  BBB         BBB
PPP              RRR      RRR             TTT                 SSS  MMM     MMM  BBB         BBB
PPP              RRR         RRR          TTT          SSSSSSSSSSS  MMM     MMM  BBBBBBBBBBBB
PPP              RRR         RRR          TTT          SSSSSSSSSSS  MMM     MMM  BBBBBBBBBBBB
PPP              RRR         RRR          TTT          SSSSSSSSSSS  MMM     MMM  BBBBBBBBBBBB
```

FILEID**DISPATCH

```
DDDDDDDD      IIIIII      SSSSSSSS  PPPPPPPP      AAAAAA    TTTTTTTTTT    CCCCCCCC  HH      HH
DDDDDDDD      IIIIII      SSSSSSSS  PPPPPPPP     AAAAAA     TTTTTTTTTT   CCCCCCCC  HH      HH
DD      DD      II      SS          PP      PP  AA      AA      TT      CC          HH      HH
DD      DD      II      SS          PP      PP  AA      AA      TT      CC          HH      HH
DD      DD      II      SS          PP      PP  AA      AA      TT      CC          HH      HH
DD      DD      II      SS          PP      PP  AA      AA      TT      CC          HH      HH
DD      DD      II        SSSSSS    PPPPPPPP    AA      AA      TT      CC          HHHHHHHHHH
DD      DD      II        SSSSSS    PPPPPPPP    AA      AA      TT      CC          HHHHHHHHHH
DD      DD      II              SS  PP          AAAAAAAAAA      TT      CC          HH      HH
DD      DD      II              SS  PP          AAAAAAAAAA      TT      CC          HH      HH
DD      DD      II              SS  PP          AA      AA      TT      CC          HH      HH      ....
DD      DD      II              SS  PP          AA      AA      TT      CC          HH      HH      ....
DDDDDDDD      IIIIII    SSSSSSSS    PP          AA      AA      TT        CCCCCCCC  HH      HH      ....
DDDDDDDD      IIIIII    SSSSSSSS    PP          AA      AA      TT        CCCCCCCC  HH      HH      ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
;     1          0001  0 MODULE DISPATCH ( %TITLE 'Print Symbiont - main dispatch routines'
;     2          0002  0          IDENT = 'V04-000',
;     3          0003  0          ADDRESSING_MODE (EXTERNAL = GENERAL)
;     4          0004  0          ) =
;     5          0005  1 BEGIN
;     6          0006  1
;     7          0007  1 !
;     8          0008  1 !**************************************************************************
;     9          0009  1 !*                                                                        *
;    10          0010  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                               *
;    11          0011  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                 *
;    12          0012  1 !*  ALL RIGHTS RESERVED.                                                   *
;    13          0013  1 !*                                                                        *
;    14          0014  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    15          0015  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    16          0016  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    17          0017  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    18          0018  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    19          0019  1 !*  TRANSFERRED.                                                           *
;    20          0020  1 !*                                                                        *
;    21          0021  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    22          0022  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    23          0023  1 !*  CORPORATION.                                                           *
;    24          0024  1 !*                                                                        *
;    25          0025  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
;    26          0026  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                *
;    27          0027  1 !*                                                                        *
;    28          0028  1 !*                                                                        *
;    29          0029  1 !**************************************************************************
;    30          0030  1 !
;    31          0031  1
;    32          0032  1 !++
;    33          0033  1 ! FACILITY:
;    34          0034  1 !       Print Symbiont.
;    35          0035  1 !
;    36          0036  1 ! ABSTRACT:
;    37          0037  1 !       This module contains the main control loop for the symbiont.
;    38          0038  1 !       PSM$DISPATCH steps through the various symbiont states and
;    39          0039  1 !       switches among the input routines.  It also calls the format
;    40          0040  1 !       and output service routines.
;    41          0041  1 !
;    42          0042  1 !       This module also contains various miscellaneous subroutines
;    43          0043  1 !       related to error handling, checkpointing, and push/pop of input
;    44          0044  1 !       routines.
;    45          0045  1 !
;    46          0046  1 ! ENVIRONMENT:
;    47          0047  1 !       VAX/VMS user mode, AST-level.
;    48          0048  1 !--
;    49          0049  1 !
;    50          0050  1 ! AUTHOR: G. Robert, CREATION DATE: 31-Aug-1982
;    51          0051  1 !
;    52          0052  1 ! MODIFIED BY:
;    53          0053  1 !
;    54          0054  1 !       3B-011  RRB3011         Rowland R. Bradley      09-Aug-1984
;    55          0055  1 !               If aligning the file and READ_COMPLETION detects EOF
;    56          0056  1 !               then send a response to job controller.  Added the
;    57          0057  1 !               test for psm$v_align in READ_COMPLETION case of
```

```
;    58          0058  1 !                   PSM$FUNCTION_DISPATCH.
;    59          0059  1 !
;    60          0060  1 !       3B-010  RRB3010         Rowland R. Bradley      27-Jul-1984
;    61          0061  1 !               Clear the suppress_output bit and the search_for_page
;    62          0062  1 !               bit on EOF (only on file service).  Also conditionally
;    63          0063  1 !               set stop_page to -1 (only when the current service is
;    64          0064  1 !               not nested).  This fixes the symbiont hang when search
;    65          0065  1 !               for page is past end of file and the /HEADER & /PAGES
;    66          0066  1 !               ill interaction.
;    67          0067  1 !
;    68          0068  1 !       3B-009  GRR3009         Gregory R. Robert       25-Jul-1984
;    69          0069  1 !               Remove the global clear of the sequence bit in print
;    70          0070  1 !               control.  This fixes the problem /header interfering
;    71          0071  1 !               with line numbers.
;    72          0072  1 !
;    73          0073  1 !       3B-008  GRR3008         Gregory R. Robert       11-Jul-1984
;    74          0074  1 !               Suppress leading carriage control for first record
;    75          0075  1 !               of implied carriage control input service.  Remove
;    76          0076  1 !               code that resets accounting totals after separation pages.
;    77          0077  1 !
;    78          0078  1 !       3B-007  GRR3007         Gregory R. Robert       16-May-1984
;    79          0079  1 !               Defend against attempted CLOSE when service routine
;    80          0080  1 !               is non-existent
;    81          0081  1 !
;    82          0082  1 !       3B-006  GRR3006         Gregory R. Robert       09-May-1984
;    83          0083  1 !               Fix call interface for user filter/format routines.
;    84          0084  1 !
;    85          0085  1 !       3B-005  GRR3005         Gregory R. Robert       29-Apr-1983
;    86          0086  1 !               FT2 bugfixes plus margins.
;    87          0087  1 !
;    88          0088  1 !       3B-004  GRR3004         Gregory R. Robert       01-Sep-1983
;    89          0089  1 !               Enabled PHY_IO so that DCS escape sequences can be
;    90          0090  1 !               written PASSALL or NOFORMAT.
;    91          0091  1 !
;    92          0092  1 !       3B-003  GRR3003         Gregory R. Robert       23-Aug-1983
;    93          0093  1 !               Bugfixes, page_setup_modules, form_setup_modules,
;    94          0094  1 !               sheet_feed, symbiont initiated pause_task and stop_stream,
;    95          0095  1 !               hangup code, read and write item services
;    96          0096  1 !
;    97          0097  1 !       3B-002  GRR3002         Gregory R. Robert       03-Aug-1983
;    98          0098  1 !               Rewrite for new design.
;    99          0099  1 !
;   100          0100  1 !       3B-001  GRR3001         Gregory R. Robert       29-Jul-1983
;   101          0101  1 !               Created new module.
;   102          0102  1 !
;   103          0103  1 !
;   104          0104  1 !**
```

```
;  106     0105  1 LIBRARY 'SYS$LIBRARY:LIB';
;  107     0106  1 REQUIRE 'LIB$:SMBDEF';
;  108     0598  1 REQUIRE 'SRC$:SMBREQ';
;  109     1055  1
;  110     1056  1 EXTERNAL ROUTINE
;  111     1057  1          PSM$ALLOCATE_DSB : NOVALUE,
;  112     1058  1          PSM$ALLOCATE_IOB : NOVALUE,
;  113     1059  1          PSM$DEALLOCATE_DSB : NOVALUE,
;  114     1060  1          SMB$INITIALIZE,
;  115     1061  1          PSM$RECEIVE_MESSAGE_AST,
;  116     1062  1          PSM$SCHEDULE_NON_AST,
;  117     1063  1          SMB$SEND_TO_JOBCTL,
;  118     1064  1          PSM$WAIT_FOR_NON_AST
;  119     1065  1          ;
;  120     1066  1
;  121     1067  1 EXTERNAL
;  122     1068  1          PSM$GL_SCBVEC    : VECTOR,                    ! SCB index table
;  123     1069  1          PSM$GL_MAXBUF    ,                            ! maximum output buffer size
;  124     1070  1          PSM$GL_USER_CTX  ,                            ! user context area size
;  125     1071  1          PSM$SRV : BLOCKVECTOR[,SRV_S_SRV, BYTE],      ! service routine table
;  126     1072  1          PSM$XLATE_ALIGN : VECTOR [,BYTE],             ! MOVTUC table for X's and 9's
;  127     1073  1          PSM$XLATE_8BIT  : VECTOR [,BYTE]              ! MOVTUC table for normal print
;  128     1074  1          ;
;  129     1075  1
;  130     1076  1 LITERAL
;  131     1077  1          EDIT_MASK = %B '110000'                       ! upcase and compact spaces and tabs
;  132     1078  1          ;
```

```
;  134          1079  1 FORWARD ROUTINE
;  135          1080  1           PSM$FUNCTION_DISPATCH   : NOVALUE,      ! main control loop
;  136          1081  1           PSM$REPORT                              ! async. event completion
;  137          1082  1           PSM$INCLUDE_MODULES     ,               ! queues modules for insertion
;  138          1083  1           PSM$PRINT                               ! initialization entry point
;  139          1084  1           PSM$STORE_ERRORS        ,               ! store errors for latter
;  140          1085  1
;  141          1086  1           ABORT_TASK              : NOVALUE,      ! aborts current file
;  142          1087  1           CARRIAGE_CONTROL        ,               ! computes carriage control
;  143          1088  1           ENQUEUE_CHECKPOINT      : NOVALUE,      ! save a checkpoint
;  144          1089  1           EXPAND_CONDITION_VECTOR : NOVALUE,      ! expand errors to text
;  145          1090  1           FIND_CHECKPOINT         ,               ! find a checkpoint
;  146          1091  1           GET_BUFFER              ,               ! get a buffer
;  147          1092  1           HANDLER                 ,               ! main signal handler
;  148          1093  1           PUTMSG_ACTION           ,               ! $PUTMSG action routine
;  149          1094  1           RESUME_SERVICE          : NOVALUE,      ! POP input routine
;  150          1095  1           SAVE_CHECKPOINT         : NOVALUE,      ! contstruct a checkpoint
;  151          1096  1           SCHEDULE_SERVICE        ,               ! schedule an input routine
;  152          1097  1           SEARCH_FOR_STRING       ,               ! look for a search string
;  153          1098  1           SUSPEND_SERVICE         : NOVALUE,      ! PUSH input routine
;  154          1099  1           STRIP_COMMA_DELIMITED_ITEM              ! parse comma separated lists
;  155          1100  1           ;
```

```
;  157          1101  1 %SBTTL 'FUNCTION_DISPATCH - Main symbiont control loop'
;  158          1102  1 ! Functional Description:
;  159          1103  1 !               Steps through symbiont states, switching among
;  160          1104  1 !               input routines and calling format/output service
;  161          1105  1 !               routines as necessary.
;  162          1106  1 !
;  163          1107  1 ! Formal Parameters:
;  164          1108  1 !               Address of a SCB (stream control block)
;  165          1109  1 !
;  166          1110  1 ! Implicit Inputs:
;  167          1111  1 !               none
;  168          1112  1 !
;  169          1113  1 ! Implicit Outputs:
;  170          1114  1 !               none
;  171          1115  1 !
;  172          1116  1 ! Returned Value:
;  173          1117  1 !               none
;  174          1118  1 !
;  175          1119  1 ! Side Effects:
;  176          1120  1 !               Asynchronous IO events may be initiated
;  177          1121  1 !--
;  178          1122  1 GLOBAL ROUTINE PSM$FUNCTION_DISPATCH (
;  179          1123  1         SCB             : REF $BBLOCK           ! stream control block address
;  180          1124  1         ) : NOVALUE =
;  181          1125  2 BEGIN
;  182          1126  2
;  183          1127  2 LITERAL
;  184          1128  2         FIRST_STATE             = 0,            ! Must be zero
;  185          1129  2         START_TASK              = FIRST_STATE,
;  186          1130  2         FIND_WORK               = 1,
;  187          1131  2         OPEN                    = 2,
;  188          1132  2         OPEN_COMPLETION         = 3,
;  189          1133  2         READ                    = 4,
;  190          1134  2         READ_COMPLETION         = 5,
;  191          1135  2         INPUT_FILTER            = 6,
;  192          1136  2         INPUT_FILTER_COMPLETION = 7,
;  193          1137  2         FORMAT                  = 8,
;  194          1138  2         FORMAT_COMPLETION       = 9,
;  195          1139  2         OUTPUT_FILTER           = 10,
;  196          1140  2         OUTPUT_FILTER_COMPLETION = 11,
;  197          1141  2         WRITE                   = 12,
;  198          1142  2         WRITE_COMPLETION        = 13,
;  199          1143  2         CLOSE                   = 14,
;  200          1144  2         CLOSE_COMPLETION        = 15,
;  201          1145  2         STOP_TASK               = 16,
;  202          1146  2         IDLE                    = 17,
;  203          1147  2         RESUME                  = 18,
;  204          1148  2         LAST_STATE              = RESUME
;  205          1149  2         ;
;  206          1150  2
;  207          1151  2 LITERAL
;  208          1152  2         CONTINUE        = 1;
;  209          1153  2
;  210          1154  2 LABEL
;  211          1155  2         CASE_STATEMENT;
;  212          1156  2
;  213          1157  2 ! For each state specify the default next_state
```

```
;  214          1158  2 !
;  215          1159  2 OWN
;  216          1160  2     NEXT_STATE : VECTOR [LAST_STATE + 1, BYTE]
;  217          1161  2         PSECT (CODE) PRESET (
;  218          1162  2             [START_TASK]                    = FIND_WORK,
;  219          1163  2             [FIND_WORK]                     = OPEN,
;  220          1164  2             [OPEN]                          = OPEN_COMPLETION,
;  221          1165  2             [OPEN_COMPLETION]               = READ,
;  222          1166  2             [READ]                          = READ_COMPLETION,
;  223          1167  2             [READ_COMPLETION]               = INPUT_FILTER,
;  224          1168  2             [INPUT_FILTER]                  = INPUT_FILTER_COMPLETION,
;  225          1169  2             [INPUT_FILTER_COMPLETION]       = FORMAT,
;  226          1170  2             [FORMAT]                        = FORMAT_COMPLETION,
;  227          1171  2             [FORMAT_COMPLETION]             = OUTPUT_FILTER,
;  228          1172  2             [OUTPUT_FILTER]                 = OUTPUT_FILTER_COMPLETION,
;  229          1173  2             [OUTPUT_FILTER_COMPLETION]      = WRITE,
;  230          1174  2             [WRITE]                         = WRITE_COMPLETION,
;  231          1175  2             [WRITE_COMPLETION]              = RESUME,
;  232          1176  2             [CLOSE]                         = CLOSE_COMPLETION,
;  233          1177  2             [CLOSE_COMPLETION]              = FIND_WORK,
;  234          1178  2             [STOP_TASK]                     = IDLE,
;  235          1179  2             [IDLE]                          = IDLE,
;  236          1180  2             [RESUME]                        = RESUME
;  237          1181  2         );
;  238          1182  2
;  239          1183  2 ! Specify expected errors that do not cause automatic task abort
;  240          1184  2 ! on a state specific basis
;  241          1185  2 !
;  242          1186  2 OWN
;  243          1187  2     EXPECTED_ERRORS : VECTOR [LAST_STATE + 1]
;  244          1188  2         PSECT (CODE) PRESET (
;  245          1189  2             [READ_COMPLETION]               = PLIT (PSM$_EOF, RMS$_EOF),
;  246          1190  2             [FORMAT_COMPLETION]             = PLIT (PSM$_BUFFEROVF, PSM$_NEWPAGE,
;  247          1191  2                                                     PSM$_ESCAPE, PSM$_SUSPEND)
;  248          1192  2         );
```

```
;  250          1193  2
;  251          1194  2 ! Advance through the symbiont states until an asynchronous service
;  252          1195  2 ! returns pending, or all output buffers are in use, or a pause is
;  253          1196  2 ! requested by the job controller
;  254          1197  2 !
;  255          1198  2 UNTIL .SCB[PSM$L_SERVICE_STATUS] EQL PSM$_PENDING
;  256          1199  2 DO
;  257          1200  2     CASE_STATEMENT:
;  258          1201  3     BEGIN
;  259          1202  3     LOCAL SERVICE : REF $BBLOCK;          ! Table entry for current input service
;  260          1203  3     LOCAL SERVICE_STATUS;                 ! Status of most recent service
;  261          1204  3     LOCAL CURRENT_STATE;                  ! Current symbiont state
;  262          1205  3
;  263          1206  3
;  264          1207  3     ! Don't do anything unless we have or can get an output buffer
;  265          1208  3     !
;  266          1209  3     IF .SCB[PSM$A_IOB] EQL 0
;  267          1210  3     THEN
;  268          1211  3         IF NOT GET_BUFFER  (.SCB)
;  269          1212  3         THEN
;  270          1213  3             RETURN;
;  271          1214  3
;  272          1215  3
;  273          1216  3     ! Locate the current input service, pickup the last
;  274          1217  3     ! service status, and initialize the next service status to success
;  275          1218  3     !
;  276          1219  3     SERVICE = PSM$SRV[.SCB[PSM$B_SERVICE_INDEX],0,0,0,0];
;  277          1220  3     SERVICE_STATUS = .SCB[PSM$L_SERVICE_STATUS];
;  278          1221  3     SCB[PSM$L_SERVICE_STATUS] = SS$_NORMAL;
;  279          1222  3
;  280          1223  3
;  281          1224  3     ! Get the current state and select the next state default
;  282          1225  3     !
;  283          1226  3     CURRENT_STATE = .SCB[PSM$B_STATE];
;  284          1227  3     SCB[PSM$B_STATE] = .NEXT_STATE[.CURRENT_STATE];
;  285          1228  3
;  286          1229  3
;  287          1230  3     ! Report any unexpected errors
;  288          1231  3     !
;  289          1232  3     IF NOT .SERVICE_STATUS
;  290          1233  3     THEN
;  291          1234  4         BEGIN
;  292          1235  4         BIND ERROR_LIST = .EXPECTED_ERRORS[.CURRENT_STATE] : VECTOR;
;  293          1236  4         LOCAL EXPECTED_ERROR : INITIAL (0);
;  294          1237  4
;  295          1238  4         ! If an expected error list is specified for the current
;  296          1239  4         ! state then loop through the list to see if the service
;  297          1240  4         ! error is expected.
;  298          1241  4         !
;  299          1242  4         IF ERROR_LIST NEQ 0
;  300          1243  4         THEN
;  301          1244  4             INCRU ERROR_INDEX TO .ERROR_LIST[-1] - 1
;  302          1245  4             DO
;  303          1246  4                 IF .SERVICE_STATUS EQL .ERROR_LIST[.ERROR_INDEX]
;  304          1247  4                 THEN
;  305          1248  5                     BEGIN
;  306          1249  5                     EXPECTED_ERROR = 1;
```

```
;   307         1250  5                     EXITLOOP;
;   308         1251  4                     END;
;   309         1252  4
;   310         1253  4          ! If an unexpected error then report it
;   311         1254  4          !
;   312         1255  4          IF NOT .EXPECTED_ERROR
;   313         1256  4          THEN
;   314         1257  4              PSM$STORE_ERRORS (.SCB, .SERVICE_STATUS);
;   315         1258  3          END;
;   316         1259  3
;   317         1260  3
;   318         1261  3      ! Dispatch to the appropriate code
;   319         1262  3      !
;   320         1263  3      CASE .CURRENT_STATE FROM FIRST_STATE TO LAST_STATE OF
;   321         1264  3          SET
;   322         1265  3
;   323         1266  3
;   324         1267  3  ! NOTE: the usual VMS/Bliss formating conventions are altered here.
;   325         1268  3  !       Each case label begins a new page and is left justifed.
;   326         1269  3  !
```

```
;  328          1270  3 [RESUME]:
;  329          1271  4 BEGIN
;  330          1272  4
;  331          1273  4 ! ++
;  332          1274  4 !
;  333          1275  4 ! RESUME handles positioning, searching, and alignment requests.
;  334          1276  4 ! The desired starting page is reached by successive approximations
;  335          1277  4 ! utilizing the POSITION_TO_KEY and REWIND service functions and the
;  336          1278  4 ! SEARCH_FOR_PAGE, SEARCH_FOR_STRING and ALIGN features of the
;  337          1279  4 ! symbiont
;  338          1280  4 !
;  339          1281  4 ! --
;  340          1282  4
;  341          1283  4 LOCAL CHECKPOINT : REF $BBLOCK;
;  342          1284  4
;  343          1285  4 ! Reset positioning and alignment controls
;  344          1286  4 !
;  345          1287  4 SCB[PSM$A_XLATE_TABLE] = PSM$XLATE_8BIT;
;  346          1288  4 SCB[PSM$V_ALIGN] = 0;
;  347          1289  4 SCB[PSM$V_SEARCH_FOR_PAGE] = 0;
;  348          1290  4 SCB[PSM$V_SEARCH_FOR_STRING] = 0;
;  349          1291  4 SCB[PSM$V_SUPPRESS_OUTPUT] = 0;
;  350          1292  4
;  351          1293  4
;  352          1294  4 ! If no start page specified then default to current page
;  353          1295  4 !
;  354          1296  4 IF .SCB[PSM$L_START_PAGE] EQL 0 THEN SCB[PSM$L_START_PAGE] = .SCB[PSM$L_PAGE];
;  355          1297  4
;  356          1298  4
;  357          1299  4 ! Look for a useable checkpoint that improves on the current page location
;  358          1300  4 !
;  359          1301  4 CHECKPOINT = FIND_CHECKPOINT (.SCB);
;  360          1302  4 IF .CHECKPOINT NEQ 0
;  361          1303  4 THEN
;  362          1304  5     BEGIN
;  363          1305  5     LOCAL KEY_DESC : VECTOR [2];
;  364          1306  5
;  365          1307  5     ! Save the checkpoint address for INPUT_FILTER_COMPLETION
;  366          1308  5     !
;  367          1309  5     SCB[PSM$A_CHECKPOINT] = .CHECKPOINT;
;  368          1310  5
;  369          1311  5
;  370          1312  5     ! Mark the next read as offset, set the new page number
;  371          1313  5     ! and cancel any outstanding input record
;  372          1314  5     !
;  373          1315  5     SCB[PSM$V_READ_OFFSET] = 1;
;  374          1316  5     SCB[PSM$L_PAGE] = .CHECKPOINT[SMBMSG$L_PAGE];
;  375          1317  5     SCB_SIZE_ (INPUT_RECORD) = 0;
;  376          1318  5
;  377          1319  5
;  378          1320  5     ! Set up the user key descriptor
;  379          1321  5     !
;  380          1322  5     KEY_DESC[0] = 4;
;  381          1323  5     KEY_DESC[1] = CHECKPOINT[SMBMSG$Q_USER_KEY];
;  382          1324  5
;  383          1325  5
;  384          1326  5     ! Request random positioning
```

```
;   385          1327  5      !
;   386          1328  5      SCB[PSM$L_SERVICE_STATUS] = BLISS (
;   387          1329  5          .SERVICE[SRV_A_SERVICE],          ! - current input service
;   388          1330  5          SCB,                              ! - SCB address by reference
;   389          1331  5          SCB[PSM$R_USER_CONTEXT_AREA],     ! - user context area
;   390          1332  5          UPLIT (PSM$K_POSITION_TO_KEY),    ! - POSITION_TO_KEY function
;   391          1333  5          KEY_DESC,                         ! - checkpoint descriptor
;   392          1334  5          0);                               ! - <not used>
;   393          1335  5
;   394          1336  5
;   395          1337  5      IF .SCB[PSM$L_SERVICE_STATUS] EQL PSM$_FUNNOTSUP
;   396          1338  5      THEN
;   397          1339  5          CODEERR_ ;      ! POSITION_TO_KEY is symetrical with GET_KEY
;   398          1340  5
;   399          1341  5
;   400          1342  5      LEAVE CASE_STATEMENT;
;   401          1343  4      END;
;   402          1344  4
;   403          1345  4
;   404          1346  4  ! If the start page is still less than the current page then rewind
;   405          1347  4  !
;   406          1348  4  IF .SCB[PSM$L_START_PAGE] LSSU .SCB[PSM$L_PAGE]
;   407          1349  4  THEN
;   408          1350  5      BEGIN
;   409          1351  5
;   410          1352  5      ! Adjust the page context and cancel any outstanding input record
;   411          1353  5      !
;   412          1354  5      SCB[PSM$L_PAGE] = 1;
;   413          1355  5      SCB[PSM$L_RECORD_NUMBER] = 0;
;   414          1356  5      SCB_SIZE_(INPUT_RECORD) = 0;
;   415          1357  5      SCB[PSM$[_CARCON] = 0;
;   416          1358  5
;   417          1359  5
;   418          1360  5      ! Request the input service to rewind
;   419          1361  5      !
;   420          1362  5      SCB[PSM$L_SERVICE_STATUS] = BLISS (
;   421          1363  5          .SERVICE[SRV_A_SERVICE],          ! - current input service
;   422          1364  5          SCB,                              ! - SCB address by reference
;   423          1365  5          SCB[PSM$R_USER_CONTEXT_AREA],     ! - user context area
;   424          1366  5          UPLIT (PSM$K_REWIND),             ! - REWIND function
;   425          1367  5          0,                                ! - <not used>
;   426          1368  5          0);                               ! - <not used>
;   427          1369  5
;   428          1370  5
;   429          1371  5      IF .SCB[PSM$L_SERVICE_STATUS] EQL PSM$_FUNNOTSUP
;   430          1372  5      THEN
;   431          1373  5          CODEERR_ ;                        ! REWIND is a required function
;   432          1374  5
;   433          1375  5      LEAVE CASE_STATEMENT;
;   434          1376  4      END;
;   435          1377  4
;   436          1378  4
;   437          1379  4  ! If the start page is still forward of the current page then start page search
;   438          1380  4  !
;   439          1381  4  IF .SCB[PSM$L_START_PAGE] GTRU .SCB[PSM$L_PAGE]
;   440          1382  4  THEN
;   441          1383  5      BEGIN
```

```
;   442          1384  5      SCB[PSM$L_STOP_PAGE] = .SCB[PSM$L_START_PAGE];
;   443          1385  5      SCB[PSM$V_SEARCH_FOR_PAGE] = 1;
;   444          1386  5      SCB[PSM$V_SUPPRESS_OUTPUT] = 1;
;   445          1387  5      SCB[PSM$B_STATE] = FORMAT;
;   446          1388  5      LEAVE CASE_STATEMENT;
;   447          1389  4      END;
;   448          1390  4
;   449          1391  4
;   450          1392  4  ! Set the stop page for string search or in case we start printing
;   451          1393  4  !
;   452          1394  4  SCB[PSM$L_STOP_PAGE] = -1;
;   453          1395  4  IF .ITEM_PRESENT_ (LAST_PAGE)
;   454          1396  4  THEN
;   455          1397  4      SCB[PSM$L_STOP_PAGE] = .SCB[PSM$L_LAST_PAGE] + 1;
;   456          1398  4
;   457          1399  4
;   458          1400  4  ! Start page reached -- initiate a string search if requested
;   459          1401  4  !
;   460          1402  4  IF TESTBITSC (ITEM_PRESENT_ (SEARCH_STRING))
;   461          1403  4  THEN
;   462          1404  5      BEGIN
;   463          1405  5      BASSEDIT (SCB[PSM$Q_SEARCH_STRING],SCB[PSM$Q_SEARCH_STRING], EDIT_MASK);
;   464          1406  5      CLEAR_STRING_ (SCB[PSM$Q_SEARCH_CONTEXT]);
;   465          1407  5      SCB[PSM$V_SEARCH_FOR_STRING] = 1;
;   466          1408  5      SCB[PSM$V_SUPPRESS_OUTPUT] = 1;
;   467          1409  5      SCB[PSM$B_STATE] = FORMAT;
;   468          1410  5      LEAVE CASE_STATEMENT;
;   469          1411  4      END;
;   470          1412  4
;   471          1413  4
;   472          1414  4  ! Positioning complete -- check for alignment
;   473          1415  4  !
;   474          1416  4  IF TESTBITSC (ITEM_PRESENT_ (ALIGNMENT_PAGES))
;   475          1417  4  THEN
;   476          1418  5      BEGIN
;   477          1419  5      SCB[PSM$V_ALIGN] = 1;
;   478          1420  5      IF .REQUEST_FLAG_ (ALIGNMENT_MASK)
;   479          1421  5      THEN
;   480          1422  5          SCB[PSM$A_XLATE_TABLE] = PSM$XLATE_ALIGN;
;   481          1423  5      SCB[PSM$L_STOP_PAGE] = .SCB[PSM$L_PAGE] + .SCB[PSM$L_ALIGNMENT_PAGES];
;   482          1424  5      SCB[PSM$B_STATE] = FORMAT;
;   483          1425  5
;   484          1426  5      ! (Since we don't alter SCB[PSM$L_START_PAGE] repositioning to
;   485          1427  5      ! the current page following alignment completion is automatic).
;   486          1428  5      !
;   487          1429  5      LEAVE CASE_STATEMENT;
;   488          1430  4      END;
;   489          1431  4
;   490          1432  4
;   491          1433  4  ! Print only one page if in sheet_feed mode
;   492          1434  4  !
;   493          1435  4  IF .$BBLOCK [SCB[PSM$L_PRINT_CONTROL], SMBMSG$V_SHEET_FEED]
;   494          1436  4  THEN
;   495          1437  4      SCB[PSM$L_STOP_PAGE] = .SCB[PSM$L_PAGE] + 1;
;   496          1438  4
;   497          1439  4
;   498          1440  4  ! Resume complete -- tell the job controller
```

```
;   499          1441  4 !
;   500          1442  4 SMB$SEND_TO_JOBCTL (
;   501          1443  4     SCB[PSM$L_STREAM_INDEX],              ! - stream number
;   502          1444  4     SCB[PSM$L_REQUEST_RESPONSE]);         ! - responding to resume or start task
;   503          1445  4
;   504          1446  4
;   505          1447  4 ! If pause at completion was requested then marks as pending
;   506          1448  4 !
;   507          1449  4 IF TESTBITSC (REQUEST_FLAG_ (PAUSE_COMPLETE))
;   508          1450  4 THEN
;   509          1451  5     BEGIN
;   510          1452  5     SCB[PSM$V_RESUME_WAIT] = 1;
;   511          1453  5     SCB[PSM$L_SERVICE_STATUS] = PSM$_PENDING;
;   512          1454  5     END
;   513          1455  4 ELSE
;   514          1456  4     SCB[PSM$B_STATE] = FORMAT;
;   515          1457  4
;   516          1458  3 END;
```

```
;   518          1459  3 [START_TASK]:
;   519          1460  4 BEGIN
;   520          1461  4
;   521          1462  4 ! Tell the job controller that START_TASK is complete and we
;   522          1463  4 ! are now printing
;   523          1464  4 !
;   524          1465  4 SMB$SEND_TO_JOBCTL (
;   525          1466  4     SCB[PSM$L_STREAM_INDEX],          ! - stream number
;   526          1467  4     SCB[PSM$L_REQUEST_RESPONSE]);     ! - responding to start task
;   527          1468  4
;   528          1469  4
;   529          1470  4 ! If pause at completion was requested then marks as pending
;   530          1471  4 !
;   531          1472  4 IF TESTBITSC (REQUEST_FLAG_ (PAUSE_COMPLETE))
;   532          1473  4 THEN
;   533          1474  5     BEGIN
;   534          1475  5     SCB[PSM$V_RESUME_WAIT] = 1;
;   535          1476  5     SCB[PSM$L_SERVICE_STATUS] = PSM$_PENDING;
;   536          1477  4     END;
;   537          1478  4
;   538          1479  3 END;
```

```
;  540          1480  3 [FIND_WORK]:
;  541          1481  4 BEGIN
;  542          1482  4
;  543          1483  4 ! If we are stopping the stream (STOP/NEXT or STOP/RESET) then stop
;  544          1484  4 ! the task
;  545          1485  4 !
;  546          1486  4 IF .SCB[PSM$V_RESET]
;  547          1487  4 THEN
;  548          1488  4     SCB[PSM$B_STATE] = STOP_TASK
;  549          1489  4 ELSE
;  550          1490  4     ! Otherwise look for an input service
;  551          1491  4     !
;  552          1492  4     IF NOT SCHEDULE_SERVICE (.SCB)
;  553          1493  4     THEN
;  554          1494  4         ! None found, cancel sheet_feed and flush the output stream
;  555          1495  4         !
;  556          1496  5         BEGIN
;  557          1497  5         $BBLOCK [SCB[PSM$L_PRINT_CONTROL], SMBMSG$V_SHEET_FEED] = 0;
;  558          1498  5         $BBLOCK [.SCB[PSM$A_IOB],IOB_V_FLUSH_PENDING] = 1;
;  559          1499  5         SCB[PSM$B_STATE] = OUTPUT_FILTER;
;  560          1500  4         END;
;  561          1501  4
;  562          1502  3 END;
```

```
;  564          1503  3 [OPEN]:
;  565          1504  4 BEGIN
;  566          1505  4
;  567          1506  4 ! If resuming a suspended service then continue at FORMAT
;  568          1507  4 !
;  569          1508  4 IF .BITVECTOR [SCB[PSM$L_SERVICE_OPEN], .SCB[PSM$B_SERVICE_INDEX]]
;  570          1509  4 THEN
;  571          1510  5     BEGIN
;  572          1511  5     SCB[PSM$B_STATE] = FORMAT;
;  573          1512  5     LEAVE CASE_STATEMENT;
;  574          1513  4     END;
;  575          1514  4
;  576          1515  4 ! Establish the default carriage control
;  577          1516  4 !
;  578          1517  4 SCB[PSM$L_FUNCTION_ARGUMENT] = PSM$K_CC_IMPLIED;
;  579          1518  4
;  580          1519  4
;  581          1520  4 ! Tell the input service to OPEN
;  582          1521  4 !
;  583          1522  4 SCB[PSM$L_SERVICE_STATUS] = BLISS (
;  584          1523  4     .SERVICE[SRV_A_SERVICE],          ! - current input service
;  585          1524  4     SCB,                              ! - SCB address by reference
;  586          1525  4     SCB[PSM$R_USER_CONTEXT_AREA],     ! - user context area
;  587          1526  4     UPLIT (PSM$K_OPEN),               ! - OPEN function
;  588          1527  4     SCB[PSM$Q_FILE_SPECIFICATION],    ! - file name
;  589          1528  4     SCB[PSM$L_FUNCTION_ARGUMENT]);    ! - receives carriage control type
;  590          1529  4
;  591          1530  3 END;
```

```
;   593          1531  3 [OPEN_COMPLETION]:
;   594          1532  4 BEGIN
;   595          1533  4
;   596          1534  4 ! If the open failed then look for more work
;   597          1535  4 !
;   598          1536  4 !
;   599          1537  4 IF NOT .SERVICE_STATUS
;   600          1538  4 THEN
;   601          1539  5     BEGIN
;   602          1540  5     SCB[PSM$B_STATE] = FIND_WORK;
;   603          1541  5     LEAVE CASE_STATEMENT;
;   604          1542  4     END;
;   605          1543  4
;   606          1544  4
;   607          1545  4 ! Mark the service OPEN and set the carriage control type
;   608          1546  4 !
;   609          1547  4 BITVECTOR [SCB[PSM$L_SERVICE_OPEN], .SCB[PSM$B_SERVICE_INDEX]] = 1;
;   610          1548  4 SCB[PSM$B_CC_TYPE] = .SCB[PSM$L_FUNCTION_ARGUMENT];
;   611          1549  4
;   612          1550  4 ! If this service is NOT a nested service then init the stop page
;   613          1551  4 ! to default of -1(end of file).
;   614          1552  4 !
;   615          1553  4 IF .SCB[PSM$B_INPUT_DEPTH] LEQ 0
;   616          1554  4 THEN
;   617          1555  4     SCB[PSM$L_STOP_PAGE] = -1;
;   618          1556  4
;   619          1557  4 ! Handle special features of main file processing including
;   620          1558  4 ! checkpoint restarts, first and last page (/PAGE=(first,last))
;   621          1559  4 ! and print flags (/FEED, /HEADER, /SPACE)
;   622          1560  4 !
;   623          1561  4 IF .SERVICE[SRV_B_SERVICE_TYPE] EQL SRV_K_FILE_SERVICE
;   624          1562  4 THEN
;   625          1563  5     BEGIN
;   626          1564  5
;   627          1565  5     ! Set the print flags
;   628          1566  5     !
;   629          1567  5     SCB[PSM$L_PRINT_FLAGS] = .SCB[PSM$L_PRINT_CONTROL];
;   630          1568  5
;   631          1569  5     ! Set up the local top and left margins (PSM$MAIN_FORMAT
;   632          1570  5     ! uses the global right and bottom margins because, with
;   633          1571  5     ! /wrap, /truncate, /feed disabled they have no effect.
;   634          1572  5     !
;   635          1573  5     SCB[PSM$L_L_MARGIN] = .SCB[PSM$L_LEFT_MARGIN];
;   636          1574  5     SCB[PSM$L_T_MARGIN] = .SCB[PSM$L_TOP_MARGIN];
;   637          1575  5
;   638          1576  5     ! Supress sequence numbers if width is too small
;   639          1577  5     !
;   640          1578  6     IF (.SCB[PSM$L_FORM_WIDTH] - .SCB[PSM$L_LEFT_MARGIN])
;   641          1579  5         - .SCB[PSM$L_RIGHT_MARGIN] LSSU 8
;   642          1580  5     THEN
;   643          1581  5         PRINT_FLAG_ (SEQUENCED) = 0;
;   644          1582  5
;   645          1583  5
;   646          1584  5     ! If restarting from a checkpoint, or if a first page was
;   647          1585  5     ! specified, then setup so that the RESUME processing will
;   648          1586  5     ! position to the correct page.
;   649          1587  5     !
```

```
;  650          1588  5      IF .ITEM_PRESENT_ (CHECKPOINT_DATA)
;  651          1589  5      THEN
;  652          1590  5          ! Checkpoint -- save it if valid
;  653          1591  6          BEGIN
;  654          1592  6          BIND CKP = .SCB_ADDR_ (CHECKPOINT_DATA) : $BBLOCK;
;  655          1593  6          IF .CKP[SMBMSG$B_CHECKPOINT_LEVEL] EQL SMBMSG$K_STRUCTURE_LEVEL
;  656          1594  6          THEN
;  657          1595  7              BEGIN
;  658          1596  7              ENQUEUE_CHECKPOINT (.SCB, SCB[PSM$Q_CHECKPOINT_DATA]);
;  659          1597  7              SCB[PSM$L_START_PAGE] = .CKP[SMBMSG$L_PAGE];
;  660          1598  7              END
;  661          1599  6          END
;  662          1600  5      ELSE
;  663          1601  5          ! /PAGE=(first_page,"")
;  664          1602  5          !
;  665          1603  5          IF .ITEM_PRESENT_ (FIRST_PAGE)
;  666          1604  5          THEN
;  667          1605  5              SCB[PSM$L_START_PAGE] = .SCB[PSM$L_FIRST_PAGE];
;  668          1606  5
;  669          1607  5      ! Flush the output stream -- positioning will be picked up
;  670          1608  5      ! after flush is complete
;  671          1609  5      !
;  672          1610  5      $BBLOCK [.SCB[PSM$A_IOB],IOB_V_FLUSH_PENDING] = 1;
;  673          1611  5      SCB[PSM$B_STATE] = OUTPUT_FILTER;
;  674          1612  4      END;
;  675          1613  3 END;
```

```
;  677          1614  3 [READ]:
;  678          1615  4 BEGIN
;  679          1616  4
;  680          1617  4 ! Initialize the user record descriptor (dynamic)
;  681          1618  4 !
;  682          1619  4 CLEAR_STRING_ (SCB[PSM$Q_USER_RECORD]);
;  683          1620  4
;  684          1621  4
;  685          1622  4 ! Quit if input service ended
;  686          1623  4 !
;  687          1624  4 IF .SCB[PSM$V_EOF] THEN LEAVE CASE_STATEMENT;
;  688          1625  4
;  689          1626  4
;  690          1627  4 ! Clear the record header field and set the new_record flag
;  691          1628  4 !
;  692          1629  4 SCB[PSM$L_RECORD_HEADER] = 0;
;  693          1630  4 SCB[PSM$V_NEW_RECORD] = 1;
;  694          1631  4
;  695          1632  4
;  696          1633  4 ! Defend against an attempt to READ a non-existent service
;  697          1634  4 !
;  698          1635  4 IF .SERVICE[SRV_A_SERVICE] EQL 0
;  699          1636  4 THEN
;  700          1637  5     BEGIN
;  701          1638  5     SERVICE_STATUS = PSM$_FUNNOTSUP;
;  702          1639  5     LEAVE CASE_STATEMENT;
;  703          1640  4     END;
;  704          1641  4
;  705          1642  4
;  706          1643  4 ! Initiate the READ
;  707          1644  4 !
;  708          1645  4 SCB[PSM$L_SERVICE_STATUS] = BLISS (
;  709          1646  4     .SERVICE[SRV_A_SERVICE],          !  - current input service
;  710          1647  4     SCB,                              !  - SCB address by reference
;  711          1648  4     SCB[PSM$R_USER_CONTEXT_AREA],     !  - user context area
;  712          1649  4     UPLIT (PSM$K_READ),               !  - READ function
;  713          1650  4     SCB[PSM$Q_USER_RECORD],           !  - quadword to receive desc
;  714          1651  4     SCB[PSM$L_RECORD_HEADER]);        !  - record header
;  715          1652  4
;  716          1653  3 END;
```

```
;  718          1654  3 [READ_COMPLETION]:
;  719          1655  4 BEGIN
;  720          1656  4
;  721          1657  4 ! Check for exceptions
;  722          1658  4 !
;  723          1659  4 IF NOT .SERVICE_STATUS
;  724          1660  4 OR .SCB[PSM$V_EOF]
;  725          1661  4 OR .SERVICE_STATUS EQL PSM$_FUNNOTSUP
;  726          1662  4 THEN
;  727          1663  5     BEGIN
;  728          1664  5
;  729          1665  5     ! Assume we will close
;  730          1666  5     !
;  731          1667  5     SCB[PSM$B_STATE] = CLOSE;
;  732          1668  5
;  733          1669  5     ! If EOF and searching for page then disable suppression and page
;  734          1670  5     ! search.
;  735          1671  5     !
;  736          1672  6     IF  (.SERVICE_STATUS EQL PSM$_EOF OR .SERVICE_STATUS EQL RMS$_EOF)
;  737          1673  6     AND (.SCB[PSM$V_SEARCH_FOR_STRING] OR .SCB[PSM$V_SEARCH_FOR_PAGE]
;  738          1674  6       OR .SCB[PSM$V_ALIGN])
;  739          1675  5     THEN
;  740          1676  6         BEGIN
;  741          1677  6         ! Only if this is a file service EOF do we wish to stop
;  742          1678  6         ! searching
;  743          1679  6         !
;  744          1680  6         IF .SERVICE[SRV_B_SERVICE_TYPE] EQL SRV_K_FILE_SERVICE
;  745          1681  6         THEN
;  746          1682  7             BEGIN
;  747          1683  7             SCB[PSM$V_SUPPRESS_OUTPUT] = 0;
;  748          1684  7             SCB[PSM$V_SEARCH_FOR_PAGE] = 0;
;  749          1685  6             END;
;  750          1686  6
;  751          1687  6         ! If EOF encountered while searching and resuming (NOT start_task)
;  752          1688  6         ! then report it and pause the thread
;  753          1689  6         !
;  754          1690  6         IF .SCB[PSM$L_REQUEST_RESPONSE] EQL SMBMSG$K_RESUME_TASK
;  755          1691  6         THEN
;  756          1692  7             BEGIN
;  757          1693  7             SMB$SEND_TO_JOBCTL (
;  758          1694  7                 SCB[PSM$L_STREAM_INDEX],          ! - stream number
;  759          1695  7                 SCB[PSM$L_REQUEST_RESPONSE],      ! - request response (resume)
;  760          1696  7                 0,                               ! - no accounting
;  761          1697  7                 0,                               ! - no checkpoint
;  762          1698  7                 0,                               ! - no device status
;  763          1699  7                 SERVICE_STATUS                   ! - report the error (eof)
;  764          1700  7                 );
;  765          1701  7             SCB[PSM$B_STATE] = RESUME;
;  766          1702  7             SCB[PSM$V_RESUME_WAIT] = 1;
;  767          1703  7             SCB[PSM$L_SERVICE_STATUS] = PSM$_PENDING;
;  768          1704  6             END;
;  769          1705  6
;  770          1706  6         LEAVE CASE_STATEMENT;
;  771          1707  5         END;
;  772          1708  4     END;
;  773          1709  4
;  774          1710  4
```

```
;  775          1711  4
;  776          1712  4
;  777          1713  4 ! Update accounting and current record number
;  778          1714  4 !
;  779          1715  4 INCREMENT_ (ACC_DATA_ (RMS_GETS));
;  780          1716  4 INCREMENT_ (SCB[PSM$C_RECORD_NUMBER]);
;  781          1717  4
;  782          1718  4
;  783          1719  4 ! If flush requested then mark the output buffer and contine
;  784          1720  4 ! at OUTPUT_FILTER
;  785          1721  4 !
;  786          1722  4 IF .SERVICE_STATUS EQL PSM$_FLUSH
;  787          1723  4 THEN
;  788          1724  5     BEGIN
;  789          1725  5     $BBLOCK [.SCB[PSM$A_IOB],IOB_V_FLUSH_PENDING] = 1;
;  790          1726  5     SCB[PSM$B_STATE] = OUTPUT_FILTER;
;  791          1727  4     END;
;  792          1728  4
;  793          1729  3 END;
```

```
;   795          1730  3 [INPUT_FILTER]:
;   796          1731  4 BEGIN
;   797          1732  4
;   798          1733  4 ! Locate the input filter
;   799          1734  4 !
;   800          1735  4 BIND FILTER = PSM$SRV[PSM$K_INPUT_FILTER,0,0,0,0] : $BBLOCK;
;   801          1736  4
;   802          1737  4
;   803          1738  4 ! If no filter then go to filter completion
;   804          1739  4 !
;   805          1740  4 IF .FILTER[SRV_A_SERVICE] EQL 0
;   806          1741  4 THEN
;   807          1742  4     LEAVE CASE_STATEMENT;
;   808          1743  4
;   809          1744  4
;   810          1745  4 ! Copy the descriptor (any class) and initialize the old one (dynamic)
;   811          1746  4 !
;   812          1747  4 COPY_QUAD_ (SCB[PSM$Q_USER_RECORD], SCB[PSM$Q_INPUT_RECORD]);
;   813          1748  4 INIT_DYN_DESC_ (SCB[PSM$Q_USER_RECORD]);
;   814          1749  4
;   815          1750  4
;   816          1751  4 ! Initiate the filter operation
;   817          1752  4 !
;   818          1753  4 SCB[PSM$L_SERVICE_STATUS] = BLISS (
;   819          1754  4     .FILTER[SRV_A_SERVICE],                ! - input filter service
;   820          1755  4     SCB,                                   ! - SCB address by reference
;   821          1756  4     SCB[PSM$R_USER_CONTEXT_AREA],          ! - user context area
;   822          1757  4     UPLIT (PSM$K_FORMAT),                  ! - FORMAT function
;   823          1758  4     SCB[PSM$Q_INPUT_RECORD],               ! - input record descriptor
;   824          1759  4     SCB[PSM$L_CARCON],                     ! - input carriage control
;   825          1760  4     SCB[PSM$Q_USER_RECORD],                ! - quadword to receive descriptor
;   826          1761  4     SCB[PSM$L_CARCON]);                    ! - output carriage control
;   827          1762  4
;   828          1763  3 END;
```

```
;  830          1764  3 [INPUT_FILTER_COMPLETION]:
;  831          1765  4 BEGIN
;  832          1766  4
;  833          1767  4 ! Initialize the input record descriptor (static)
;  834          1768  4 !
;  835          1769  4 STR$ANALYZE_SDESC_R1 (
;  836          1770  4     SCB[PSM$Q_USER_RECORD]                        ! Input record descriptor
;  837          1771  4     ,
;  838          1772  4     VECTOR [SCB[PSM$Q_INPUT_RECORD],0],           ! R0 -> size
;  839          1773  4     VECTOR [SCB[PSM$Q_INPUT_RECORD],1]);          ! R1 -> address
;  840          1774  4
;  841          1775  4
;  842          1776  4 ! If the first byte of the record was used for carriage control
;  843          1777  4 ! (eg. FORTRAN) then remove it from the record descriptor
;  844          1778  4 !
;  845          1779  4 IF CARRIAGE_CONTROL (.SCB) EQL PSM$K_FIRST_CHAR_USED
;  846          1780  4 THEN
;  847          1781  5     BEGIN
;  848          1782  5     DECREMENT_ (SCB_SIZE_ (INPUT_RECORD));
;  849          1783  5     INCREMENT_ (SCB_ADDR_ (INPUT_RECORD));
;  850          1784  4     END;
;  851          1785  4
;  852          1786  4
;  853          1787  4 ! If this is an offset read (that is, one that is to begin in the
;  854          1788  4 ! middle of a record) then adjust the record descriptor by the offset
;  855          1789  4 ! value from the checkpoint.
;  856          1790  4 !
;  857          1791  4 IF TESTBITSC (SCB[PSM$V_READ_OFFSET])
;  858          1792  4 THEN
;  859          1793  5     BEGIN
;  860          1794  5     BIND CHECKPOINT = .SCB[PSM$A_CHECKPOINT] : $BBLOCK;
;  861          1795  5     SCB_SIZE_ (INPUT_RECORD) = .SCB_SIZE_ (INPUT_RECORD)
;  862          1796  5         - .CHECKPOINT[SMBMSG$W_OFFSET];
;  863          1797  5     SCB_ADDR_ (INPUT_RECORD) = .SCB_ADDR_ (INPUT_RECORD)
;  864          1798  5         + .CHECKPOINT[SMBMSG$W_OFFSET];
;  865          1799  5     SCB[PSM$L_CARCON] = .CHECKPOINT[SMBMSG$L_CARCON];
;  866          1800  5     SCB[PSM$L_RECORD_NUMBER] = .CHECKPOINT[SMBMSG$L_RECORD_NUMBER];
;  867          1801  4     END;
;  868          1802  4
;  869          1803  4
;  870          1804  3 END;
```

```
;  872          1805  3 [FORMAT]:
;  873          1806  4 BEGIN
;  874          1807  4
;  875          1808  4 ! Locate the main format routine
;  876          1809  4 !
;  877          1810  4 BIND FILTER  = PSM$SRV[PSM$K_MAIN_FORMAT,0,0,0,0] : $BBLOCK;
;  878          1811  4
;  879          1812  4
;  880          1813  4 ! Initiate the FORMAT function
;  881          1814  4 !
;  882          1815  4 SCB[PSM$L_SERVICE_STATUS] = BLISS (
;  883          1816  4     .FILTER[SRV_A_SERVICE],                ! - format service
;  884          1817  4     SCB,                                   ! - SCB address by reference
;  885          1818  4     SCB[PSM$R_USER_CONTEXT_AREA],          ! - user context area
;  886          1819  4     UPLIT (PSM$K_FORMAT),                  ! - FORMAT function
;  887          1820  4     SCB[PSM$Q_INPUT_RECORD],               ! - input record descriptor
;  888          1821  4     SCB[PSM$L_CARCON],                     ! - input carriage control
;  889          1822  4     SCB[PSM$Q_OUTPUT_BUFFER],              ! - output buffer descriptor
;  890          1823  4     0);                                    ! - unused function argument
;  891          1824  4
;  892          1825  3 END;
```

```
;   894          1826  3 [FORMAT_COMPLETION]:
;   895          1827  4 BEGIN
;   896          1828  4
;   897          1829  4 ! If succesfull then block multiple input records into a single
;   898          1830  4 ! output buffer by continuing at READ.
;   899          1831  4 !
;   900          1832  4 IF .SERVICE_STATUS
;   901          1833  4 THEN
;   902          1834  5     BEGIN
;   903          1835  5     SCB[PSM$B_STATE] = READ;
;   904          1836  5     LEAVE CASE_STATEMENT;
;   905          1837  4     END;
;   906          1838  4
;   907          1839  4
;   908          1840  4 ! If starting an escape sequence then mark escape in progress.
;   909          1841  4 ! Insure that there are at least two bytes remaining in the output
;   910          1842  4 ! buffer to allow two-byte escape sequences to be assembled.
;   911          1843  4 !
;   912          1844  4 IF .SERVICE_STATUS EQL PSM$_ESCAPE
;   913          1845  4 THEN
;   914          1846  5     BEGIN
;   915          1847  5     SCB[PSM$B_ESCAPE_STATE] = 0;
;   916          1848  5     SCB[PSM$V_ESCAPE_IN_PROGRESS] = 1;
;   917          1849  5     SCB[PSM$B_STATE] = FORMAT;
;   918          1850  5
;   919          1851  5     ! If there are at least two output bytes remaining then continue
;   920          1852  5     ! at FORMAT, else write the buffer.
;   921          1853  5     !
;   922          1854  5     IF .SCB_SIZE_ (OUTPUT_BUFFER) GTRU 2
;   923          1855  5     THEN
;   924          1856  5         SCB[PSM$B_STATE] = FORMAT;
;   925          1857  5     LEAVE CASE_STATEMENT;
;   926          1858  4     END;
;   927          1859  4
;   928          1860  4
;   929          1861  4 ! See if format service requesting suspension (OSC)
;   930          1862  4 !
;   931          1863  4 IF .SERVICE_STATUS EQL PSM$_SUSPEND
;   932          1864  4 THEN
;   933          1865  5     BEGIN
;   934          1866  5     SUSPEND_SERVICE (.SCB);
;   935          1867  5     SCB[PSM$B_STATE] = FIND_WORK;
;   936          1868  5     LEAVE CASE_STATEMENT;
;   937          1869  4     END;
;   938          1870  4
;   939          1871  4
;   940          1872  4 ! If output buffer full then write it
;   941          1873  4 !
;   942          1874  4 IF .SERVICE_STATUS EQL PSM$_BUFFEROVF
;   943          1875  4 THEN
;   944          1876  4     LEAVE CASE_STATEMENT;
;   945          1877  4
;   946          1878  4
;   947          1879  4 ! Must be a new page
;   948          1880  4 !
;   949          1881  4 IF .SERVICE_STATUS NEQ PSM$_NEWPAGE THEN CODEERR_ ;
;   950          1882  4
```

```
;  951          1883  4
;  952          1884  4 ! New page -- save a checkpoint if 32 pages have passed or if
;  953          1885  4 ! we are stopping on this page
;  954          1886  4 !
;  955          1887  4 IF (.SCB[PSM$L_PAGE] AND %B '11111') EQL 0
;  956          1888  4 OR .SCB[PSM$L_PAGE] GEQU .SCB[PSM$L_STOP_PAGE]
;  957          1889  4 THEN
;  958          1890  4     SAVE_CHECKPOINT (.SCB);
;  959          1891  4
;  960          1892  4
;  961          1893  4 ! If we are stopping on this page then flush the output stream
;  962          1894  4 ! and reset the "new page" trigger
;  963          1895  4 !
;  964          1896  4 IF .SCB[PSM$L_PAGE] GEQU .SCB[PSM$L_STOP_PAGE]
;  965          1897  4 THEN
;  966          1898  5     BEGIN
;  967          1899  5     $BBLOCK [.SCB[PSM$A_IOB],IOB_V_FLUSH_PENDING] = 1;
;  968          1900  5     SCB[PSM$L_LINE] = 0;
;  969          1901  5     LEAVE CASE_STATEMENT;
;  970          1902  4     END;
;  971          1903  4
;  972          1904  4
;  973          1905  4 ! Check for string search -- if the output buffer is not empty
;  974          1906  4 ! then force a buffer write
;  975          1907  4 !
;  976          1908  4 IF .SCB[PSM$V_SEARCH_FOR_STRING]
;  977          1909  4 THEN
;  978          1910  5     BEGIN
;  979          1911  5     BIND IOB = .SCB[PSM$A_IOB] : $BBLOCK;
;  980          1912  5     IF .SCB_SIZE_ (OUTPUT_BUFFER) NEQ .DESC_SIZE_ (IOB[IOB_0_BUFFER])
;  981          1913  5     THEN
;  982          1914  5         ! Reset the new page trigger and force a buffer write
;  983          1915  5         !
;  984          1916  6         BEGIN
;  985          1917  6         SCB[PSM$L_LINE] = 0;
;  986          1918  6         LEAVE CASE_STATEMENT;
;  987          1919  5         END;
;  988          1920  4     END;
;  989          1921  4
;  990          1922  4
;  991          1923  4 ! Check for page headers and/or page setup
;  992          1924  4 !
;  993          1925  4 IF .PRINT_FLAG_ (PAGE_HEADER) THEN SERVICE_LIST_ (PAGE_HEADER) = 1;
;  994          1926  4 IF .SCB_SIZE_ (PAGE_SETUP_MODULES) NEQ 0
;  995          1927  4 OR .PSM$SRV[PSM$K_PAGE_SETUP, SRV_V_USER_SUPPLIED]
;  996          1928  4 THEN
;  997          1929  4     SERVICE_LIST_ (PAGE_SETUP) = 1;
;  998          1930  4
;  999          1931  4
; 1000          1932  4 ! If page headers or setup required then suspend current input service
; 1001          1933  4 ! and continue at FIND_WORK.
; 1002          1934  4 !
; 1003          1935  4 IF .SERVICE_LIST_ (PAGE_HEADER)
; 1004          1936  4 OR .SERVICE_LIST_ (PAGE_SETUP)
; 1005          1937  4 THEN
; 1006          1938  5     BEGIN
; 1007          1939  5     SUSPEND_SERVICE (.SCB);
```

```
; 1008          1940  5      SCB[PSM$B_STATE] = FIND_WORK;
; 1009          1941  5      LEAVE CASE_STATEMENT;
; 1010          1942  4      END;
; 1011          1943  4
; 1012          1944  4
; 1013          1945  4  ! If new page with no side effects then continue at FORMAT
; 1014          1946  4  ! else go to next state (output_filter)
; 1015          1947  4  !
; 1016          1948  4  IF NOT .SCB[PSM$V_SEARCH_FOR_STRING]
; 1017          1949  4  THEN
; 1018          1950  4      SCB[PSM$B_STATE] = FORMAT;
; 1019          1951  4
; 1020          1952  3  END;
```

```
;  1022          1953  3 [OUTPUT_FILTER]:
;  1023          1954  4 BEGIN
;  1024          1955  4
;  1025          1956  4 ! Locate the output filter service, the output block, and the output record
;  1026          1957  4 !
;  1027          1958  4 BIND FILTER = PSM$SRV[PSM$K_OUTPUT_FILTER,0,0,0,0] : $BBLOCK;
;  1028          1959  4 BIND IOB = .SCB[PSM$A_IOB] : $BBLOCK;
;  1029          1960  4 BIND IOBREC = IOB[IOB_Q_RECORD] : VECTOR;
;  1030          1961  4
;  1031          1962  4
;  1032          1963  4 ! Clear the old record descriptor (any class) and set it to
;  1033          1964  4 ! the size of the blocked record buffer (static)
;  1034          1965  4 !
;  1035          1966  4 CLEAR_STRING_ (IOBREC);
;  1036          1967  4 IOBREC[1] = .DESC_ADDR_ (IOB[IOB_Q_BUFFER]);
;  1037          1968  4 IOBREC[0] = .SCB_ADDR_ (OUTPUT_BUFFER) - .DESC_ADDR_ (IOB[IOB_Q_BUFFER]);
;  1038          1969  4 IF .IOBREC[0] GTRU .DESC_SIZE_ (IOB[IOB_Q_BUFFER]) THEN CODEERR_ ;
;  1039          1970  4
;  1040          1971  4
;  1041          1972  4 ! If no output filter then bypass service call
;  1042          1973  4 !
;  1043          1974  4 IF .FILTER[SRV_A_SERVICE] EQL 0
;  1044          1975  4 THEN
;  1045          1976  4     LEAVE CASE_STATEMENT;
;  1046          1977  4
;  1047          1978  4
;  1048          1979  4 ! Copy the output record decriptor (static) and reinitialize it (dynamic)
;  1049          1980  4 !
;  1050          1981  4 COPY_QUAD_ (IOBREC, SCB[PSM$Q_OUTPUT_BUFFER]);
;  1051          1982  4 INIT_DYN_DESC_ (IOBREC);
;  1052          1983  4
;  1053          1984  4
;  1054          1985  4 ! Call the output filter service
;  1055          1986  4 !
;  1056          1987  4 SCB[PSM$L_SERVICE_STATUS] = BLISS (
;  1057          1988  4     .FILTER[SRV_A_SERVICE],                 !  - output filter service
;  1058          1989  4     SCB,                                    !  - SCB address by reference
;  1059          1990  4     SCB[PSM$R_USER_CONTEXT_AREA],           !  - user context area
;  1060          1991  4     UPLIT (PSM$K_FORMAT),                   !  - FORMAT function
;  1061          1992  4     SCB[PSM$Q_OUTPUT_BUFFER],               !  - input to filter
;  1062          1993  4     0,                                      !  - unused function argument
;  1063          1994  4     IOBREC,                                 !  - output from filter
;  1064          1995  4     0);                                     !  - unused function argument
;  1065          1996  4
;  1066          1997  3 END;
```

```
;  1068          1998  3 [OUTPUT_FILTER_COMPLETION]:
;  1069          1999  4 BEGIN
;  1070          2000  4
;  1071          2001  4 ! Locate the OUTPUT block
;  1072          2002  4 !
;  1073          2003  4 BIND IOB = .SCB[PSM$A_IOB] : $BBLOCK;
;  1074          2004  4
;  1075          2005  4
;  1076          2006  4 ! Check for string search
;  1077          2007  4 !
;  1078          2008  4 IF .SCB[PSM$V_SEARCH_FOR_STRING]
;  1079          2009  4 THEN
;  1080          2010  4     IF SEARCH_FOR_STRING (.SCB, SCB[PSM$Q_SEARCH_STRING], IOB[IOB_Q_RECORD])
;  1081          2011  4     THEN
;  1082          2012  4
;  1083          2013  4         ! String found -- release the output buffer, set the start
;  1084          2014  4         ! page, and continue at RESUME
;  1085          2015  4         !
;  1086          2016  5         BEGIN
;  1087          2017  5         INSERT_TAIL_ (.SCB[PSM$A_IOB], SCB[PSM$Q_BUFFER_QUEUE]);
;  1088          2018  5         SCB[PSM$A_IOB] = 0;
;  1089          2019  5         SCB[PSM$L_START_PAGE] = .SCB[PSM$L_PAGE];
;  1090          2020  5
;  1091          2021  5         ! If sitting at top of page then we really want to restart at
;  1092          2022  5         ! the previous page
;  1093          2023  5         !
;  1094          2024  5         IF .SCB[PSM$L_LINE] LEQU 1
;  1095          2025  5         AND .SCB[PSM$L_COLUMN] LEQU 1
;  1096          2026  5         AND .SCB[PSM$L_PAGE] GTRU 1
;  1097          2027  5         THEN
;  1098          2028  6             DECREMENT_ (SCB[PSM$L_START_PAGE])
;  1099          2029  5         ELSE
;  1100          2030  5             ! Mid-page: force RESUME to reposition by fibbing about
;  1101          2031  5             ! current page
;  1102          2032  5             !
;  1103          2033  5             INCREMENT_ (SCB[PSM$L_PAGE]);
;  1104          2034  5         SCB[PSM$B_STATE] = RESUME;
;  1105          2035  5         LEAVE CASE_STATEMENT;
;  1106          2036  4         END;
;  1107          2037  4
;  1108          2038  3 END;
```

```
;  1110          2039  3 [WRITE]:
;  1111          2040  4 BEGIN
;  1112          2041  4
;  1113          2042  4 ! Locate the output block and the output service routine
;  1114          2043  4 !
;  1115          2044  4 BIND IOB = .SCB[PSM$A_IOB] : $BBLOCK;
;  1116          2045  4 BIND OUTPUT = PSM$SRV[PSM$K_OUTPUT,0,0,0,0] : $BBLOCK;
;  1117          2046  4
;  1118          2047  4
;  1119          2048  4 ! Establish the default function as WRITE
;  1120          2049  4 !
;  1121          2050  4 LOCAL FUNCTION : INITIAL (PSM$K_WRITE);
;  1122          2051  4
;  1123          2052  4
;  1124          2053  4 ! Check for /PASSALL or buffer marked passall (DCS's)
;  1125          2054  4 !
;  1126          2055  4 IF .PRINT_FLAG_ (PASSALL)
;  1127          2056  4 OR .IOB[IOB_V_PASSALL]
;  1128          2057  4 THEN
;  1129          2058  4     FUNCTION = PSM$K_WRITE_NOFORMAT;
;  1130          2059  4
;  1131          2060  4
;  1132          2061  4 ! Check for write suppression (searching)
;  1133          2062  4 !
;  1134          2063  4 IF .SCB[PSM$V_SUPPRESS_OUTPUT]
;  1135          2064  4 THEN
;  1136          2065  4     FUNCTION = PSM$K_WRITE_SUPPRESSED
;  1137          2066  4 ELSE
;  1138          2067  4     INCREMENT_ (ACC_DATA_ (QIO_PUTS));
;  1139          2068  4
;  1140          2069  4
;  1141          2070  4 ! Initiate the WRITE function
;  1142          2071  4 !
;  1143          2072  4 SCB[PSM$L_SERVICE_STATUS] = BLISS (
;  1144          2073  4     .OUTPUT[SRV_A_SERVICE],              !  - write service
;  1145          2074  4     SCB[PSM$A_IOB],                      !  - IOB address by reference
;  1146          2075  4     SCB[PSM$R_USER_CONTEXT_AREA],        !  - user context area
;  1147          2076  4     FUNCTION,                            !  - WRITE or WRITE_SUPPRESSED function
;  1148          2077  4     IOB[IOB_Q_RECORD],                   !  - record desc
;  1149          2078  4     0);                                  !  - <not used>
;  1150          2079  4
;  1151          2080  4
;  1152          2081  4 ! Disconnect the IOB from the SCB
;  1153          2082  4 !
;  1154          2083  4 SCB[PSM$A_IOB] = 0;
;  1155          2084  4
;  1156          2085  4
;  1157          2086  4 ! Asynchronous?
;  1158          2087  4 !
;  1159          2088  4 IF .SCB[PSM$L_SERVICE_STATUS] EQL PSM$_PENDING
;  1160          2089  4 THEN
;  1161          2090  5     BEGIN
;  1162          2091  5
;  1163          2092  5     ! Yes: don't wait for completion unless we are flushing the output stream
;  1164          2093  5     ! Either way, PSM$REPORT will release the IOB
;  1165          2094  5     !
;  1166          2095  5     IF NOT .IOB[IOB_V_FLUSH_PENDING]
```

```
;  1167          2096  5      THEN
;  1168          2097  6          BEGIN
;  1169          2098  6          SCB[PSM$L_SERVICE_STATUS] = SS$_NORMAL;
;  1170          2099  6          SCB[PSM$B_STATE] = FORMAT;
;  1171          2100  5          END;
;  1172          2101  5      END
;  1173          2102  4  ELSE
;  1174          2103  5      BEGIN
;  1175          2104  5      ! Synchronous return; release the IOB
;  1176          2105  5      !
;  1177          2106  5      INSERT_TAIL_ (IOB[IOB_Q_QLINKS], SCB[PSM$Q_BUFFER_QUEUE]);
;  1178          2107  5
;  1179          2108  5      ! If successful, and not flushing, then continue at FORMAT
;  1180          2109  5      !
;  1181          2110  5      IF .SCB[PSM$L_SERVICE_STATUS] EQL SS$_NORMAL
;  1182          2111  5      AND NOT .IOB[IOB_V_FLUSH_PENDING]
;  1183          2112  5      THEN
;  1184          2113  5          SCB[PSM$B_STATE] = FORMAT;
;  1185          2114  4      END;
;  1186          2115  4
;  1187          2116  3  END;
```

```
; 1189          2117  3 [WRITE_COMPLETION]:
; 1190          2118  4 BEGIN
; 1191          2119  4
; 1192          2120  4 ! If the IO failed then the error has already been stored and task abort begun.
; 1193          2121  4 ! continue at READ.
; 1194          2122  4 !
; 1195          2123  4 IF NOT .SCB[PSM$L_SERVICE_STATUS]
; 1196          2124  4 THEN
; 1197          2125  5     BEGIN
; 1198          2126  5     SCB[PSM$B_STATE] = READ;
; 1199          2127  5     LEAVE CASE_STATEMENT;
; 1200          2128  4     END;
; 1201          2129  4
; 1202          2130  4
; 1203          2131  4 ! The write was successful -- we are here because the output stream
; 1204          2132  4 ! is being flushed for one of:
; 1205          2133  4 !
; 1206          2134  4 !       1) Last page reached (PRINT /PAGE=last)
; 1207          2135  4 !       2) Job controller requested pause (STOP /QUEUE)
; 1208          2136  4 !       3) A page search operation has completed
; 1209          2137  4 !       4) An alignment operation has completed (START /QUEUE /ALIGN=pages)
; 1210          2138  4 !       5) We are in sheet feed mode (DEFINE /FORM /SHEET_FEED)
; 1211          2139  4 !
; 1212          2140  4 ! Respond based on why we are flushing
; 1213          2141  4 !
; 1214          2142  4
; 1215          2143  4 ! If pausing then mark the stream pending and respond to the job controller
; 1216          2144  4 !
; 1217          2145  4 IF .SCB[PSM$L_REQUEST_RESPONSE] EQL SMBMSG$K_PAUSE_TASK
; 1218          2146  4 THEN
; 1219          2147  5     BEGIN
; 1220          2148  5     SMB$SEND_TO_JOBCTL (
; 1221          2149  5         SCB[PSM$L_STREAM_INDEX],        ! - stream number
; 1222          2150  5         SCB[PSM$L_REQUEST_RESPONSE]);   ! - request response
; 1223          2151  5     SCB[PSM$V_RESUME_WAIT] = 1;
; 1224          2152  5     SCB[PSM$L_SERVICE_STATUS] = PSM$_PENDING;
; 1225          2153  5     LEAVE CASE_STATEMENT;
; 1226          2154  4     END;
; 1227          2155  4
; 1228          2156  4
; 1229          2157  4 ! If searching for a string then continue formatting
; 1230          2158  4 !
; 1231          2159  4 IF .SCB[PSM$V_SEARCH_FOR_STRING]
; 1232          2160  4 THEN
; 1233          2161  5     BEGIN
; 1234          2162  5     SCB[PSM$B_STATE] = FORMAT;
; 1235          2163  5     LEAVE CASE_STATEMENT;
; 1236          2164  4     END;
; 1237          2165  4
; 1238          2166  4
; 1239          2167  4 ! If searching for a page or aligning then go to next state (resume)
; 1240          2168  4 !
; 1241          2169  4 IF .SCB[PSM$V_ALIGN]
; 1242          2170  4 OR .SCB[PSM$V_SEARCH_FOR_PAGE]
; 1243          2171  4 THEN
; 1244          2172  4     LEAVE CASE_STATEMENT;
; 1245          2173  4
```

```
;  1246          2174  4
;  1247          2175  4 ! Sheet feeding?
;  1248          2176  4 !
;  1249          2177  4 IF .$BBLOCK [SCB[PSM$L_PRINT_CONTROL], SMBMSG$V_SHEET_FEED]
;  1250          2178  4 AND NOT .SCB[PSM$V_SUPPRESS_OUTPUT]
;  1251          2179  4 THEN
;  1252          2180  5     BEGIN
;  1253          2181  5     LOCAL DEVICE_STATUS;
;  1254          2182  5     DEVICE_STATUS = .SCB[PSM$L_DEVICE_STATUS] OR SMBMSG$M_PAUSE_TASK;
;  1255          2183  5     SMB$SEND_TO_JOBCTL (
;  1256          2184  5         SCB[PSM$L_STREAM_INDEX],          ! - stream number
;  1257          2185  5         UPLIT (SMBMSG$K_TASK_STATUS),     ! - request response
;  1258          2186  5         0,                                ! - no accounting
;  1259          2187  5         0,                                ! - no checkpoint
;  1260          2188  5         DEVICE_STATUS                     ! - device status (paused)
;  1261          2189  5         );
;  1262          2190  5     SCB[PSM$V_RESUME_WAIT] = 1;
;  1263          2191  5     SCB[PSM$L_SERVICE_STATUS] = PSM$_PENDING;
;  1264          2192  5     LEAVE CASE_STATEMENT;
;  1265          2193  4     END;
;  1266          2194  4
;  1267          2195  4 IF .SCB[PSM$L_SERVICE_LIST] EQL 0 THEN SCB[PSM$B_STATE] = STOP_TASK
;  1268          2196  4 ELSE
;  1269          2197  4     IF .ITEM_PRESENT  (LAST_PAGE)
;  1270          2198  4     AND .SCB[PSM$L_PAGE] GTRU .SCB[PSM$L_LAST_PAGE]
;  1271          2199  4     THEN
;  1272          2200  4         SCB[PSM$B_STATE] = CLOSE;
;  1273          2201  4
;  1274          2202  3 END;
```

```
;  1276          2203  3 [CLOSE]:
;  1277          2204  4 BEGIN
;  1278          2205  4
;  1279          2206  4 ! Defend against an attempt to CLOSE a non-existent service
;  1280          2207  4 !
;  1281          2208  4 IF .SERVICE[SRV_A_SERVICE] EQL 0
;  1282          2209  4 THEN
;  1283          2210  4     LEAVE CASE_STATEMENT;
;  1284          2211  4
;  1285          2212  4
;  1286          2213  4 ! Initiate the CLOSE function for the current input routine
;  1287          2214  4 !
;  1288          2215  4 SCB[PSM$L_SERVICE_STATUS] = BLISS (
;  1289          2216  4     .SERVICE[SRV_A_SERVICE],          !  - current input service
;  1290          2217  4     SCB,                              !  - SCB address by reference
;  1291          2218  4     SCB[PSM$R_USER_CONTEXT_AREA],     !  - user context area
;  1292          2219  4     UPLIT (PSM$K_CLOSE),              !  - CLOSE function
;  1293          2220  4     0,                                !  - <not used>
;  1294          2221  4     0);                               !  - <not used>
;  1295          2222  4
;  1296          2223  3 END;
```

```
; 1298          2224  3 [CLOSE_COMPLETION]:
; 1299          2225  4 BEGIN
; 1300          2226  4
; 1301          2227  4 ! Mark the service closed
; 1302          2228  4 !
; 1303          2229  4 BITVECTOR [SCB[PSM$L_SERVICE_OPEN], .SCB[PSM$B_SERVICE_INDEX]] = 0;
; 1304          2230  4
; 1305          2231  4
; 1306          2232  4 ! If this was a forced EOF and input was nested then pass the
; 1307          2233  4 ! abort flag to the next service, else clear it
; 1308          2234  4 !
; 1309          2235  4 IF TESTBITSC (SCB[PSM$V_EOF])
; 1310          2236  4 THEN
; 1311          2237  4     IF .SCB[PSM$B_INPUT_DEPTH] NEQ 0
; 1312          2238  4     THEN
; 1313          2239  4         SCB[PSM$V_EOF] = 1;
; 1314          2240  4
; 1315          2241  3 END;
```

```
; 1317          2242  3 [STOP_TASK]:
; 1318          2243  4 BEGIN
; 1319          2244  4
; 1320          2245  4 ! A stream is "active" if its queue is started.  It is busy if it
; 1321          2246  4 ! is currently processing a task.
; 1322          2247  4 !
; 1323          2248  4 LOCAL
; 1324          2249  4         ACTIVE_STREAMS  : INITIAL (0),          ! number of active streams
; 1325          2250  4         BUSY_STREAMS    : INITIAL (0)           ! number of busy streams
; 1326          2251  4         ;
; 1327          2252  4
; 1328          2253  4 ! Clear any pending input service routines from the service list and
; 1329          2254  4 ! reset the busy and reset flags.
; 1330          2255  4 !
; 1331          2256  4 SCB[PSM$L_SERVICE_LIST] = 0;
; 1332          2257  4 SCB[PSM$V_BUSY] = 0;
; 1333          2258  4 SCB[PSM$V_RESET] = 0;
; 1334          2259  4
; 1335          2260  4 ! If the job controller did not request an abort then we respond
; 1336          2261  4 ! with the asynchronous TASK_COMPLETE message.  Otherwise we respond
; 1337          2262  4 ! with the current contents of REQUEST_RESPONSE which is presumably
; 1338          2263  4 ! STOP_TASK or RESET_TASK.
; 1339          2264  4 !
; 1340          2265  4 IF .SCB[PSM$L_REQUEST_RESPONSE] EQL SMBMSG$K_START_TASK
; 1341          2266  4 OR .SCB[PSM$L_REQUEST_RESPONSE] EQL SMBMSG$K_RESUME_TASK
; 1342          2267  4 THEN
; 1343          2268  4     SCB[PSM$L_REQUEST_RESPONSE] = SMBMSG$K_TASK_COMPLETE;
; 1344          2269  4
; 1345          2270  4
; 1346          2271  4 ! Notify the job controller
; 1347          2272  4 !
; 1348          2273  4 SMB$SEND_TO_JOBCTL (
; 1349          2274  4         SCB[PSM$L_STREAM_INDEX],        ! - stream number
; 1350          2275  4         SCB[PSM$L_REQUEST_RESPONSE],    ! - responding to ...
; 1351          2276  4         SCB[PSM$Q_ACCOUNTING_DATA],     ! - accounting data
; 1352          2277  4         0,                              ! - no checkpoint
; 1353          2278  4         SCB[PSM$L_DEVICE_STATUS],       ! - device status
; 1354          2279  4         SCB[PSM$T_CONDITION_AREA]       ! - errors if any
; 1355          2280  4         );
; 1356          2281  4
; 1357          2282  4 ! Now scan to see if there are any active or busy streams
; 1358          2283  4 !
; 1359          2284  4 INCR I TO PSM$K_MAXSTREAMS - 1
; 1360          2285  4 DO
; 1361          2286  5     BEGIN
; 1362          2287  5     BIND SCBPTR  = .PSM$GL_SCBVEC [.I] : $BBLOCK;
; 1363          2288  5     IF SCBPTR NEQ 0
; 1364          2289  5     THEN
; 1365          2290  6         BEGIN
; 1366          2291  6         IF .SCBPTR[PSM$V_ACTIVE]
; 1367          2292  6         THEN
; 1368          2293  6             INCREMENT  (ACTIVE_STREAMS);
; 1369          2294  6         IF .SCBPTR[PSM$V_BUSY]
; 1370          2295  6         THEN
; 1371          2296  6             INCREMENT_ (BUSY_STREAMS);
; 1372          2297  6         END
; 1373          2298  4     END;
```

```
;  1374          2299  4
;  1375          2300  4
;  1376          2301  4 ! If no active streams then exit
;  1377          2302  4 !
;  1378          2303  4 IF .ACTIVE_STREAMS EQL 0
;  1379          2304  4 THEN
;  1380          2305  4     $EXIT (CODE = SS$_NORMAL OR STS$M_INHIB_MSG);
;  1381          2306  4
;  1382          2307  4
;  1383          2308  4 ! If no busy streams then purge the working set
;  1384          2309  4 !
;  1385          2310  4 IF .BUSY_STREAMS EQL 0
;  1386          2311  4 THEN
;  1387          2312  4     $PURGWS (INADR=UPLIT (0, %X '7FFFFFFF'));
;  1388          2313  4
;  1389          2314  4
;  1390          2315  3 END;
```

```
;  1392          2316  3 [IDLE]:
;  1393          2317  4 BEGIN
;  1394          2318  4
;  1395          2319  4 ! If a reset has occured then contine at STOP_TASK
;  1396          2320  4 !
;  1397          2321  4 IF .SCB[PSM$V_RESET]
;  1398          2322  4 THEN
;  1399          2323  4     SCB[PSM$B_STATE] = STOP_TASK
;  1400          2324  4 ELSE
;  1401          2325  4     RETURN;
;  1402          2326  4
;  1403          2327  3 END;
```

```
; 1405            2328  3
; 1406            2329  3 ! Usual formatting conventions resume here
; 1407            2330  3 !
; 1408            2331  3         TES;    ! End of case table
; 1409            2332  2     END;        ! End of CASE_STATEMENT block
; 1410            2333  1 END;            ! End of PSMSFUNCTION_DISPATCH routine


                                                                .TITLE  DISPATCH Print Symbiont - main dispatch routine
;                                                                       s
                                                                .IDENT  \V04-000\

                                                                .PSECT  CODE,NOWRT,2

0F  12  0D  0C  0B  0A  09  08  07  06  05  04  03  02  01  00000 NEXT_STATE:
                                                                .BYTE   1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, -        ;
                                                                        13, 18, 15, 1, 17, 17, 18                       ;
                                                            00013       .BLKB   1                                       ;
                                                 00000002   00014       .LONG   2                                       ;
                                                 00000000G  00018 P.AAA: .LONG   PSM$_EOF                                ;
                                                 0001827A   0001C       .LONG   98938                                   ;
                                                 00000004   00020       .LONG   4                                       ;
                    00000000G 00000000G 00000000G 00000000G 00024 P.AAB: .LONG   PSM$_BUFFEROVF, PSM$_NEWPAGE, -        ;
                                                                        PSM$_ESCAPE, PSM$_SUSPEND                       ;
                                                       00# 00034 EXPECTED_ERRORS:
                                                                .BYTE   0[20]                                           ;
                                                 00000000'  00048       .ADDRESS P.AAA                                  ;
                                                       00#  0004C       .BYTE   0[12]                                   ;
                                                 00000000'  00058       .ADDRESS P.AAB                                  ;
                                                            0005C       .BLKB   36                                      ;
                                                 00000007   00080 P.AAC: .LONG   7                                       ;
                                                 00000008   00084 P.AAD: .LONG   8                                       ;
                                                 00000004   00088 P.AAE: .LONG   4                                       ;
                                                 00000005   0008C P.AAF: .LONG   5                                       ;
                                                 00000003   00090 P.AAG: .LONG   3                                       ;
                                                 00000003   00094 P.AAH: .LONG   3                                       ;
                                                 00000003   00098 P.AAI: .LONG   3                                       ;
                                                 00000009   0009C P.AAJ: .LONG   9                                       ;
                                                 00000002   000A0 P.AAK: .LONG   2                                       ;
                    7FFFFFFF                     00000000   000A4 P.AAL: .LONG   0, 2147483647                          ;

                                                                .EXTRN  BAS$EDIT, LBR$CLOSE
                                                                .EXTRN  LBR$GET_RECORD, LBR$INI_CONTROL
                                                                .EXTRN  LBR$LOOKUP_KEY, LBR$OPEN
                                                                .EXTRN  LBR$RET_RMSSTV, LBR$SET_LOCATE
                                                                .EXTRN  LIB$TRIM_FILESPEC
                                                                .EXTRN  LIB$GET_VM, LIB$FREE_VM
                                                                .EXTRN  STR$ANALYZE_SDESC
                                                                .EXTRN  STR$ANALYZE_SDESC_R1
                                                                .EXTRN  STR$APPEND, STR$CONCAT
                                                                .EXTRN  STR$COPY_DX, STR$COPY_R
                                                                .EXTRN  STR$FREE1_DX, STR$FREE1_DX_R4
                                                                .EXTRN  STR$GET1_DX, STR$LEFT
                                                                .EXTRN  STR$PREFIX, STR$RIGHT
                                                                .EXTRN  PSM$S_HANGUP_DISPATCH_ENTRY
                                                                .EXTRN  PSM$_BUFFEROVF, PSM$_EOF
                                                                .EXTRN  PSM$_ESCAPE, PSM$_FLUSH
```

```
                                                .EXTRN  PSM$_FUNNOTSUP, PSM$_INVITMCOD
                                                .EXTRN  PSM$_INVVMSOSC, PSM$_MODNOTFND
                                                .EXTRN  PSM$_NEWPAGE, PSM$_NOFILEID
                                                .EXTRN  PSM$_OSCTOOLON, PSM$_PENDING
                                                .EXTRN  PSM$_SUSPEND, PSM$_TOOMANYLEV
                                                .EXTRN  SMB$_INVSTMNBR, SMB$_INVSTRLEV
                                                .EXTRN  SMB$_NOMOREITEMS
                                                .EXTRN  PSM$ALLOCATE_DSB
                                                .EXTRN  PSM$ALLOCATE_IOB
                                                .EXTRN  PSM$DEALLOCATE_DSB
                                                .EXTRN  SMB$INITIALIZE, PSM$RECEIVE_MESSAGE_AST
                                                .EXTRN  PSM$SCHEDULE_NON_AST
                                                .EXTRN  SMB$SEND_TO_JOBCTL
                                                .EXTRN  PSM$WAIT_FOR_NON_AST
                                                .EXTRN  PSM$GL_SCBVEC, PSM$GL_MAXBUF
                                                .EXTRN  PSM$GL_USER_CTX
                                                .EXTRN  PSM$SRV, PSM$XLATE_ALIGN
                                                .EXTRN  PSM$XLATE_8BIT, SYS$EXIT
                                                .EXTRN  SYS$PURGWS

                                  0FFC 00000    .ENTRY  PSM$FUNCTION_DISPATCH, Save R2,R3,R4,R5,R6,-; 1122
                                                        R7,R8,R9,R10,R11
                        5E          14  C2 00002         SUBL2   #20, SP
                        52      04  AC  D0 00005 1$:     MOVL    SCB, R2                             1198
                        54    0220  C2  9E 00009         MOVAB   544(R2), R4
            00000000G   8F          64  D1 0000E         CMPL    (R4), #PSM$_PENDING
                                    01  12 00015         BNEQ    2$
                                    04  00017         RET
                        58    01AC  C2  9E 00018 2$:     MOVAB   428(R2), R8                         1209
                                    68  D5 0001D         TSTL    (R8)
                                    0B  12 0001F         BNEQ    3$
                                    52  DD 00021         PUSHL   R2                                  1211
                  0000V CF          01  FB 00023         CALLS   #1, GET_BUFFER
                        01          50  E8 00028         BLBS    R0, 3$
                                    04  0002B         RET
                        5B    027D  C2  9E 0002C 3$:     MOVAB   637(R2), R11                        1219
                        50          6B  9A 00031         MOVZBL  (R11), R0
                        50          10  C4 00034         MULL2   #16, R0
                        55 00000000G0040  9E 00037         MOVAB   PSM$SRV[R0], SERVICE
                        6E          64  D0 0003F         MOVL    (R4), SERVICE_STATUS                1220
                        64          01  D0 00042         MOVL    #1, (R4)                            1221
                        56    02A7  C2  9E 00045         MOVAB   679(R2), R6                         1226
                        53          66  9A 0004A         MOVZBL  (R6), CURRENT_STATE
                        66  FF02 CF43  90 0004D         MOVB    NEXT_STATE[CURRENT_STATE], (R6)     1227
                        57          6E  D0 00053         MOVL    SERVICE_STATUS, R7                  1232
                        33          57  E8 00056         BLBS    R7, 8$
                        51  FF2A CF43  D0 00059         MOVL    EXPECTED_ERRORS[CURRENT_STATE], R1  1235
                                    59  D4 0005F         CLRL    EXPECTED_ERROR
                                    51  D5 00061         TSTL    R1                                  1242
                                    1B  13 00063         BEQL    7$
                  5A    FC    A1      01  C3 00065         SUBL3   #1, -4(R1), R10                   1244
                                    50  D4 0006A         CLRL    ERROR_INDEX
                                    0D  11 0006C         BRB     6$
                      6140          57  D1 0006E 4$:     CMPL    R7, (R1)[ERROR_INDEX]               1246
                                    05  12 00072         BNEQ    5$
                        59          01  D0 00074         MOVL    #1, EXPECTED_ERROR                  1249
                                    07  11 00077         BRB     7$                                  1248
```

```
                                                50  D6 00079 5$:    INCL    ERROR_INDEX                                     ; 1246
                                             5A     50  D1 0007B 6$:    CMPL    ERROR_INDEX, R10                                ;
                                                    EE  1B 0007E        BLEQU   4$                                              ;
                                             09     59  E8 00080 7$:    BLBS    EXPECTED_ERROR, 8$                              ; 1255
                                         0084    8F  BB 00083        PUSHR   #^M<R2,R7>                                      ; 1257
                                   0000V CF      02  FB 00087        CALLS   #2, PSM$STORE_ERRORS                            ;
                                       12        00  53  CF 0008C 8$:    CASEL   CURRENT_STATE, #0, #18                          ; 1263
                01FF                01DF            01C5            01AD    00090 9$:    .WORD   27$-9$,-                        ;
                0394                0358            02D9            0282    00098        28$-9$,-                                ;
                0557                04D2            0403            03DD    000A0        30$-9$,-                                ;
                06B9                069F            0619            05A8    000A8        32$-9$,-                                ;
                                    0026            074C            06D6    000B0        39$-9$,-                                ;
                                                                                         44$-9$,-                                ;
                                                                                         52$-9$,-                                ;
                                                                                         53$-9$,-                                ;
                                                                                         56$-9$,-                                ;
                                                                                         58$-9$,-                                ;
                                                                                         75$-9$,-                                ;
                                                                                         79$-9$,-                                ;
                                                                                         83$-9$,-                                ;
                                                                                         89$-9$,-                                ;
                                                                                         103$-9$,-                               ;
                                                                                         107$-9$,-                               ;
                                                                                         111$-9$,-                               ;
                                                                                         119$-9$,-                               ;
                                                                                         10$-9$                                  ;
                                0244    C2 00000000G    00  9E 000B6 10$:    MOVAB   PSM$XLATE_8BIT, 580(R2)                 ; 1287
                                       53        10    A2  9E 000BF        MOVAB   16(R2), R3                              ; 1288
                                       63      7001    8F  AA 000C3        BICW2   #28673, (R3)                            ; 1291
                                               0224    C2  D5 000C8        TSTL    548(R2)                                 ; 1296
                                                       07  12 000CC        BNEQ    11$                                     ;
                                0224    C2      01EC    C2  D0 000CE        MOVL    492(R2), 548(R2)                        ;
                                                       52  DD 000D5 11$:    PUSHL   R2                                      ; 1301
                                0000V   CF              01  FB 000D7        CALLS   #1, FIND_CHECKPOINT                     ;
                                                       50  D5 000DC        TSTL    CHECKPOINT                              ; 1302
                                                       35  13 000DE        BEQL    12$                                     ;
                                0190    C2              50  D0 000E0        MOVL    CHECKPOINT, 400(R2)                     ; 1309
                                  01    A3              02  88 000E5        BISB2   #2, 1(R3)                               ; 1315
                                01EC    C2      08      A0  D0 000E9        MOVL    8(CHECKPOINT), 492(R2)                  ; 1316
                                             0260    C2  B4 000EF        CLRW    608(R2)                                 ; 1317
                                  0C    AE              04  D0 000F3        MOVL    #4, KEY_DESC                            ; 1322
                                  10    AE        10    A0  9E 000F7        MOVAB   16(R0), KEY_DESC+4                      ; 1323
                                                       7E  D4 000FC        CLRL    -(SP)                                   ; 1331
                                                 10    AE  9F 000FE        PUSHAB  KEY_DESC                                ;
                                                FECF    CF  9F 00101        PUSHAB  P.AAC                                   ; 1332
                                                02D0    C2  9F 00105        PUSHAB  720(R2)                                 ; 1331
                                                   04    AC  9F 00109        PUSHAB  SCB                                     ; 1328
                                00      B5              05  FB 0010C        CALLS   #5, @0(SERVICE)                         ; 1331
                                               64       50  D0 00110        MOVL    R0, (R4)                                ;
                                                       34  11 00113        BRB     13$                                     ; 1337
                                               53    04    AC  D0 00115 12$:    MOVL    SCB, R3                                 ; 1348
                                01EC    C3      0224    C3  D1 00119        CMPL    548(R3), 492(R3)                        ;
                                                       48  1E 00120        BGEQU   15$                                     ;
                                01EC    C3              01  D0 00122        MOVL    #1, 492(R3)                             ; 1354
                                             026C    C3  D4 00127        CLRL    620(R3)                                 ; 1355
                                             0260    C3  B4 0012B        CLRW    608(R3)                                 ; 1356
                                             0278    C3  D4 0012F        CLRL    632(R3)                                 ; 1357
```

```
                                   7E   7C 00133           CLRQ     -(SP)                                    ; 1365
                         FE9F      CF   9F 00135           PUSHAB   P.AAD                                    ; 1366
                         02D0      C3   9F 00139           PUSHAB   720(R3)                                  ; 1365
                           04      AC   9F 0013D           PUSHAB   SCB                                      ; 1362
            00     B5              05   FB 00140           CALLS    #5, @0(SERVICE)                          ; 1365
          0220     C3              50   D0 00144           MOVL     R0, 544(R3)                              ;
                   50      04      AC   D0 00149 13$:      MOVL     SCB, R0                                  ; 1371
     00000000G     8F    0220      C0   D1 0014D           CMPL     544(R0), #PSM$_FUNNOTSUP                 ;
                                   0F   12 00156           BNEQ     14$                                      ;
                                   01   DD 00158           PUSHL    #1                                       ; 1372
                     01061154      8F   DD 0015A           PUSHL    #17174868                                ;
     00000000G     00              02   FB 00160           CALLS    #2, LIB$STOP                             ;
                              FE9B 31   00167 14$:         BRW      1$                                       ; 1375
                   53      04      AC   D0 0016A 15$:      MOVL     SCB, R3                                  ; 1381
                   55    01EC      C3   9E 0016E           MOVAB    492(R3), R5                              ;
                   65    0224      C3   D1 00173           CMPL     548(R3), (R5)                            ;
                                   0D   1B 00178           BLEQU    16$                                      ;
          0228     C3    0224      C3   D0 0017A           MOVL     548(R3), 552(R3)                         ; 1384
            11     A3              10   88 00181           BISB2    #16, 17(R3)                              ; 1385
                                   5C   11 00185           BRB      20$                                      ; 1386
                   52    0228      C3   9E 00187 16$:      MOVAB    552(R3), R2                              ; 1394
                   62              01   CE 0018C           MNEGL    #1, (R2)                                 ;
                   54    01B0      C3   9E 0018F           MOVAB    432(R3), R4                              ; 1395
 06                64              1B   E1 00194           BBC      #27, (R4), 17$                           ;
 62       00B8     C3              01   C1 00198           ADDL3    #1, 184(R3), (R2)                        ; 1397
 48                64              32   E5 0019E 17$:      BBCC     #50, (R4), 21$                           ; 1402
                                   30   DD 001A2           PUSHL    #48                                      ; 1405
                         014C      C3   9F 001A4           PUSHAB   332(R3)                                  ;
                         014C      C3   9F 001A8           PUSHAB   332(R3)                                  ;
     00000000G     00              03   FB 001AC           CALLS    #3, BAS$EDIT                             ;
                   01    0213      C3   91 001B3           CMPB     531(R3), #1                              ; 1406
                                   11   1A 001B8           BGTRU    18$                                      ;
                   50    0210      C3   9E 001BA           MOVAB    528(R3), R0                              ;
                   60 020E0000     8F   D0 001BF           MOVL     #34471936, (R0)                          ;
                           04      A0   D4 001C6           CLRL     4(R0)                                    ;
                                   14   11 001C9           BRB      19$                                      ;
                   50              53   D0 001CB 18$:      MOVL     R3, R0                                   ;
                         0210      C0   B5 001CE           TSTW     528(R0)                                  ;
                                   0B   13 001D2           BEQL     19$                                      ;
                         0210      C3   9F 001D4           PUSHAB   528(R3)                                  ;
     00000000G     00              01   FB 001D8           CALLS    #1, STR$FREE1_DX                         ;
            11     A3              20   88 001DF 19$:      BISB2    #32, 17(R3)                              ; 1407
            11     A3      40      8F   88 001E3 20$:      BISB2    #64, 17(R3)                              ; 1408
                                   4B   11 001E8           BRB      25$                                      ; 1409
 19                64              04   E5 001EA 21$:      BBCC     #4, (R4), 23$                            ; 1416
            10     A3              01   88 001EE           BISB2    #1, 16(R3)                               ; 1419
                   09    0140      C3   E9 001F2           BLBC     320(R3), 22$                             ; 1420
          0244     C3 00000000G    00   9E 001F7           MOVAB    PSM$XLATE_ALIGN, 580(R3)                 ; 1422
 62                65      2C      A3   C1 00200 22$:      ADDL3    44(R3), (R5), (R2)                       ; 1423
                                   2E   11 00205           BRB      25$                                      ; 1424
 04       0124     C3              05   E1 00207 23$:      BBC      #5, 292(R3), 24$                         ; 1435
 62                65              01   C1 0020D           ADDL3    #1, (R5), (R2)                           ; 1437
                         0144      C3   9F 00211 24$:      PUSHAB   324(R3)                                  ; 1444
                         022C      C3   9F 00215           PUSHAB   556(R3)                                  ; 1443
     00000000G     00              02   FB 00219           CALLS    #2, SMB$SEND_TO_JOBCTL                   ; 1444
 0F       0140     C3              01   E5 00220           BBCC     #1, 320(R3), 25$                         ; 1449
            11     A3              08   88 00226           BISB2    #8, 17(R3)                               ; 1452
```

```
                     0220   C3 00000000G   8F   D0 0022A          MOVL    #PSM$_PENDING, 544(R3)                  ; 1453
                                           05   11 00233          BRB     26$                                     ; 1449
                     02A7   C3             08   90 00235 25$:     MOVB    #8, 679(R3)                             ; 1456
                                       FDC8   31 0023A 26$:      BRW     1$                                      ; 1263
                                  0144   C2   9F 0023D 27$:      PUSHAB  324(R2)                                 ; 1467
                                  022C   C2   9F 00241           PUSHAB  556(R2)                                 ; 1466
                00000000G   00           02   FB 00245          CALLS   #2, SMB$SEND_TO_JOBCTL                  ; 1467
                E8   0140   C2           01   E5 0024C          BBCC    #1, 320(R2), 26$                        ; 1472
                                       0472   31 00252          BRW     93$                                     ; 1475
                03     11   A2           02   E1 00255 28$:     BBC     #2, 17(R2), 29$                         ; 1486
                                       0584   31 0025A          BRW     120$                                    ;
                                         52   DD 0025D 29$:     PUSHL   R2                                      ; 1492
                     0000V  CF           01   FB 0025F          CALLS   #1, SCHEDULE_SERVICE                    ;
                            D3           50   E8 00264          BLBS    R0, 26$                                 ;
                     0124   C2           20   8A 00267          BICB2   #32, 292(R2)                            ; 1497
                                       016C   31 0026C          BRW     50$                                     ; 1498
                            50           6B   9A 0026F 30$:     MOVZBL  (R11), R0                               ; 1508
                03   021C   C2           50   E1 00272          BBC     R0, 540(R2), 31$                        ;
                                       045A   31 00278          BRW     95$                                     ;
                     01A8   C2           02   D0 0027B 31$:     MOVL    #2, 424(R2)                             ; 1517
                                  01A8   C2   9F 00280          PUSHAB  424(R2)                                 ; 1528
                                  0098   C2   9F 00284          PUSHAB  152(R2)                                 ; 1527
                                  FD50   CF   9F 00288          PUSHAB  P.AAE                                   ; 1526
                                       04AA   31 0028C          BRW     104$                                    ; 1525
                            03           57   E8 0028F 32$:     BLBS    R7, 33$                                 ; 1537
                                       02BF   31 00292          BRW     72$                                     ;
                            50           6B   9A 00295 33$:     MOVZBL  (R11), R0                               ; 1547
                00   021C   C2           50   E2 00298          BBSS    R0, 540(R2), 34$                        ;
                     027C   C2      01A8   C2   90 0029E 34$:     MOVB    424(R2), 636(R2)                        ; 1548
                                  02A5   C2   95 002A5          TSTB    677(R2)                                 ; 1553
                                         05   14 002A9          BGTR    35$                                     ;
                     0228   C2             01   CE 002AB          MNEGL   #1, 552(R2)                             ; 1555
                            01       0C     A5   91 002B0 35$:     CMPB    12(SERVICE), #1                         ; 1561
                                           84   12 002B4          BNEQ    26$                                     ;
                     0204   C2      0124   C2   D0 002B6          MOVL    292(R2), 516(R2)                        ; 1567
                     01BC   C2      00BC   C2   D0 002BD          MOVL    188(R2), 444(R2)                        ; 1573
                     0230   C2      0164   C2   D0 002C4          MOVL    356(R2), 560(R2)                        ; 1574
           50        008C   C2      00BC   C2   C3 002CB          SUBL3   188(R2), 140(R2), R0                    ; 1578
                            50      0148   C2   C2 002D3          SUBL2   328(R2), R0                             ; 1579
                            08             50   D1 002D8          CMPL    R0, #8                                  ;
                                           05   1E 002DB          BGEQU   36$                                     ;
                     0204   C2             10   8A 002DD          BICB2   #16, 516(R2)                            ; 1581
                                  01B0   C2   95 002E2 36$:     TSTB    432(R2)                                 ; 1588
                                         1C   18 002E6          BGEQ    37$                                     ;
                            53     40     A2   D0 002E8          MOVL    64(R2), R3                              ; 1592
                            01     01     A3   91 002EC          CMPB    1(R3), #1                               ; 1593
                                         1D   12 002F0          BNEQ    38$                                     ;
                                   3C     A2   9F 002F2          PUSHAB  60(R2)                                  ; 1596
                                          52   DD 002F5          PUSHL   R2                                      ;
                     0000V  CF             02   FB 002F7          CALLS   #2, ENQUEUE_CHECKPOINT                  ;
                     0224   C2      08     A3   D0 002FC          MOVL    8(R3), 548(R2)                          ; 1597
                                           0B   11 00302          BRB     38$                                     ; 1593
                            06     01B2   C2   E9 00304 37$:     BLBC    434(R2), 38$                            ; 1603
                     0224   C2      74     A2   D0 00309          MOVL    116(R2), 548(R2)                        ; 1605
                                       00C9   31 0030F 38$:      BRW     50$                                     ; 1610
                            01     0273   C2   91 00312 39$:     CMPB    627(R2), #1                             ; 1619
                                           11   1A 00317          BGTRU   40$                                     ;
```

```
                                50      0270   C2  9E 00319          MOVAB   624(R2), R0                                     ;
                                60 020E0000    8F  D0 0031E          MOVL    #34471936, (R0)                                 ;
                                        04     A0  D4 00325          CLRL    4(R0)                                           ;
                                               14  11 00328          BRB     41$                                             ;
                                50             52  D0 0032A 40$:     MOVL    R2, R0                                          ;
                                        0270   C0  B5 0032D          TSTW    624(R0)                                         ;
                                               0B  13 00331          BEQL    41$                                             ;
                                        0270   C2  9F 00333          PUSHAB  624(R2)                                         ;
                    00000000G   00             01  FB 00337          CALLS   #1, STR$FREE1_DX                                ;
                14          10  A2             02  E0 0033E 41$:     BBS     #2, 16(R2), 42$                                 ; 1624
                                        0268   C2  D4 00343          CLRL    616(R2)                                         ; 1629
                            10  A2          80  8F  88 00347          BISB2   #128, 16(R2)                                    ; 1630
                                               65  D5 0034C          TSTL    (SERVICE)                                       ; 1635
                                               0A  12 0034E          BNEQ    43$                                             ;
                                6E 00000000G   8F  D0 00350          MOVL    #PSM$_FUNNOTSUP, SERVICE_STATUS                 ; 1638
                                          FCAB  31  00357 42$:       BRW     1$                                              ; 1639
                                        0268   C2  9F 0035A 43$:     PUSHAB  616(R2)                                         ; 1651
                                        0270   C2  9F 0035E          PUSHAB  624(R2)                                         ; 1650
                                        FC7A   CF  9F 00362          PUSHAB  P.AAF                                           ; 1649
                                             03D0  31  00366          BRW     104$                                            ; 1648
                                0E             57  E9 00369 44$:     BLBC    R7, 45$                                         ; 1659
                09          10  A2             02  E0 0036C          BBS     #2, 16(R2), 45$                                 ; 1660
                    00000000G   8F             57  D1 00371          CMPL    R7, #PSM$_FUNNOTSUP                             ; 1661
                                               50  12 00378          BNEQ    49$                                             ;
                                66             0E  90 0037A 45$:     MOVB    #14, (R6)                                       ; 1667
                    00000000G   8F             57  D1 0037D          CMPL    R7, #PSM$_EOF                                   ; 1672
                                               09  13 00384          BEQL    46$                                             ;
                    0001827A    8F             57  D1 00386          CMPL    R7, #98938                                      ;
                                               3B  12 0038D          BNEQ    49$                                             ;
                09          11  A2             05  E0 0038F 46$:     BBS     #5, 17(R2), 47$                                 ; 1673
                04          11  A2             04  E0 00394          BBS     #4, 17(R2), 47$                                 ;
                                2D         10  A2  E9 00399          BLBC    16(R2), 49$                                     ; 1674
                                01         0C  A5  91 0039D 47$:     CMPB    12(SERVICE), #1                                 ; 1680
                                               05  12 003A1          BNEQ    48$                                             ;
                            11  A2          50  8F  8A 003A3          BICB2   #80, 17(R2)                                     ; 1684
                                03     0144    C2  D1 003A8 48$:     CMPL    324(R2), #3                                     ; 1690
                                               A8  12 003AD          BNEQ    42$                                             ;
                                               5E  DD 003AF          PUSHL   SP                                              ; 1695
                                               7E  7C 003B1          CLRQ    -(SP)                                           ;
                                               7E  D4 003B3          CLRL    -(SP)                                           ;
                                        0144   C2  9F 003B5          PUSHAB  324(R2)                                         ;
                                        022C   C2  9F 003B9          PUSHAB  556(R2)                                         ; 1694
                    00000000G   00             06  FB 003BD          CALLS   #6, SMB$SEND_TO_JOBCTL                          ; 1695
                                66             12  90 003C4          MOVB    #18, (R6)                                       ; 1701
                                             02FD  31  003C7          BRW     93$                                             ; 1702
                                        0286   C2  D6 003CA 49$:     INCL    646(R2)                                         ; 1715
                                        026C   C2  D6 003CE          INCL    620(R2)                                         ; 1716
                    00000000G   8F             6E  D1 003D2          CMPL    SERVICE_STATUS, #PSM$_FLUSH                     ; 1722
                                               0A  12 003D9          BNEQ    51$                                             ;
                                50             68  D0 003DB 50$:     MOVL    (R8), R0                                        ; 1725
                            2C  A0             02  88 003DE          BISB2   #2, 44(R0)                                      ;
                                66             0A  90 003E2          MOVB    #10, (R6)                                       ; 1726
                                             FC1D  31  003E5 51$:     BRW     1$                                              ; 1263
                                53 00000000G   00  D0 003E8 52$:     MOVL    FILTER, R3                                      ; 1740
                                               F4  13 003EF          BEQL    51$                                             ;
                                51      0260   C2  9E 003F1          MOVAB   608(R2), R1                                     ; 1747
                                50      0270   C2  9E 003F6          MOVAB   624(R2), R0                                     ;
```

```
                          61                    60  7D  003FB          MOVQ    (R0), (R1)
                          60 020E0000           8F  D0  003FE          MOVL    #34471936, (R0)                 ; 1748
                                04              A0  D4  00405          CLRL    4(R0)
                              0278              C2  9F  00408          PUSHAB  632(R2)                         ; 1761
                                                50  DD  0040C          PUSHL   R0
                              0278              C2  9F  0040E          PUSHAB  632(R2)                         ; 1759
                                                51  DD  00412          PUSHL   R1                              ; 1761
                              FBCC              CF  9F  00414          PUSHAB  P.AAG                           ; 1757
                              02D0              C2  9F  00418          PUSHAB  720(R2)                         ; 1756
                                04              AC  9F  0041C          PUSHAB  SCB                             ; 1753
                          63                    07  FB  0041F          CALLS   #7, (R3)                        ; 1761
                                                6C  11  00422          BRB     57$
                          50        0270        C2  9E  00424  53$:     MOVAB   624(R2), R0                     ; 1770
                                000000000G      00  16  00429          JSB     STR$ANALYZE_SDESC_R1            ; 1773
                          53        0260        C2  9E  0042F          MOVAB   608(R2), R3                     ; 1772
                          63                    50  7D  00434          MOVQ    R0, (R3)
                                                52  DD  00437          PUSHL   R2                              ; 1779
                    0000V CF                    01  FB  00439          CALLS   #1, CARRIAGE_CONTROL
                          03                    50  D1  0043E          CMPL    R0, #3
                                                05  12  00441          BNEQ    54$
                                                63  B7  00443          DECW    (R3)                            ; 1782
                                    04          A3  D6  00445          INCL    4(R3)                           ; 1783
                 9B       10      A2            09  E5  00448  54$:     BBCC    #9, 16(R2), 51$                 ; 1791
                          50        0190        C2  D0  0044D          MOVL    400(R2), R0                     ; 1794
                          63          02        A0  A2  00452          SUBW2   2(R0), (R3)                     ; 1796
                          51          02        A0  3C  00456          MOVZWL  2(R0), R1                       ; 1798
                       04 A3                    51  C0  0045A          ADDL2   R1, 4(R3)
                     0278 C2            04      A0  D0  0045E          MOVL    4(R0), 632(R2)                  ; 1799
                     026C C2            0C      A0  D0  00464          MOVL    12(R0), 620(R2)                 ; 1800
                                      FB98      31  00467A 55$:        BRW     1$                              ; 1263
                          50 00000000G          00  D0  0046D  56$:     MOVL    FILTER, R0                      ; 1816
                                                7E  D4  00474          CLRL    -(SP)                           ; 1822
                                    01E0        C2  9F  00476          PUSHAB  480(R2)
                                    0278        C2  9F  0047A          PUSHAB  632(R2)                         ; 1821
                                    0260        C2  9F  0047E          PUSHAB  608(R2)                         ; 1820
                                    FB62        CF  9F  00482          PUSHAB  P.AAH                           ; 1819
                                    02D0        C2  9F  00486          PUSHAB  720(R2)                         ; 1818
                                      04        AC  9F  0048A          PUSHAB  SCB                             ; 1815
                          60                    07  FB  0048D          CALLS   #7, (R0)                        ; 1822
                                              02B1  31  00490  57$:     BRW     105$
                          03                    57  E9  00493  58$:     BLBC    R7, 59$                         ; 1832
                                              0213  31  00496          BRW     90$
 00000000G                8F                   57  D1  00499  59$:     CMPL    R7, #PSM$_ESCAPE                ; 1844
                                                15  12  004A0          BNEQ    60$
                                    02A3        C2  94  004A2          CLRB    675(R2)                         ; 1847
                       10 A2                    08  88  004A6          BISB2   #8, 16(R2)                      ; 1848
                          66                    08  90  004AA          MOVB    #8, (R6)                        ; 1849
                          02        01E0        C2  B1  004AD          CMPW    480(R2), #2                     ; 1854
                                                B6  1B  004B2          BLEQU   55$
                                              021E  31  004B4          BRW     95$                             ; 1856
 00000000G                8F                   57  D1  004B7  60$:     CMPL    R7, #PSM$_SUSPEND               ; 1863
                                                03  12  004BE          BNEQ    61$
                                              008A  31  004C0          BRW     71$
 00000000G                8F                   57  D1  004C3  61$:     CMPL    R7, #PSM$_BUFFEROVF             ; 1874
                                                9E  13  004CA          BEQL    55$
 00000000G                8F                   57  D1  004CC          CMPL    R7, #PSM$_NEWPAGE               ; 1881
                                                0F  13  004D3          BEQL    62$
```

```
                                   01  DD 004D5        PUSHL   #1                                    ;
                         01061154  8F  DD 004D7        PUSHL   #17174868                             ;
                00000000G  00      02  FB 004DD        CALLS   #2, LIB$STOP                          ;
                           1F  01EC C2  93 004E4 62$:  BITB    492(R2), #31                          ; 1887
                                   09  13 004E9        BEQL    63$                                   ;
                     0228  C2  01EC C2  D1 004EB       CMPL    492(R2), 552(R2)                      ; 1888
                                   07  1F 004F2        BLSSU   64$                                   ;
                                   52  DD 004F4 63$:   PUSHL   R2                                    ; 1890
                     0000V  CF     01  FB 004F6        CALLS   #1, SAVE_CHECKPOINT                   ;
                     0228  C2  01EC C2  D1 004FB 64$:  CMPL    492(R2), 552(R2)                      ; 1896
                                   09  1F 00502        BLSSU   65$                                   ;
                           50      68  D0 00504        MOVL    (R8), R0                              ; 1899
                       2C  A0      02  88 00507        BISB2   #2, 44(R0)                            ;
                                   10  11 0050B        BRB     66$                                   ; 1900
            11         11  A2      05  E1 0050D 65$:   BBC     #5, 17(R2), 67$                       ; 1908
                           50      68  D0 00512        MOVL    (R8), R0                              ; 1911
                       1C  A0  01E0 C2  B1 00515       CMPW    480(R2), 28(R0)                       ; 1912
                                   06  13 0051B        BEQL    67$                                   ;
                               01CB C2  D4 0051D 66$:  CLRL    456(R2)                               ; 1917
                                   34  11 00521        BRB     73$                                   ; 1918
            05       0204  C2      01  E1 00523 67$:   BBC     #1, 516(R2), 68$                      ; 1925
                     0218  C2      04  88 00529        BISB2   #4, 536(R2)                           ;
                               00DC C2  B5 0052E 68$:  TSTW    220(R2)                               ; 1926
                                   08  12 00532        BNEQ    69$                                   ;
            05 00000000G  00      01  E1 00534         BBC     #1, PSM$SRV+24, 70$                   ; 1927
                     0218  C2      02  88 0053C 69$:   BISB2   #2, 536(R2)                           ; 1929
            06       0218  C2      02  E0 00541 70$:   BBS     #2, 536(R2), 71$                      ; 1935
            0D       0218  C2      01  E1 00547        BBC     #1, 536(R2), 74$                      ; 1936
                                   52  DD 0054D 71$:   PUSHL   R2                                    ; 1939
                     0000V  CF     01  FB 0054F        CALLS   #1, SUSPEND_SERVICE                   ;
                           66      01  90 00554 72$:   MOVB    #1, (R6)                              ; 1940
                               FAAB  31 00557 73$:     BRW     1$                                    ; 1941
            F8         11  A2      05  E0 0055A 74$:   BBS     #5, 17(R2), 73$                       ; 1948
                               0173  31 0055F          BRW     95$                                   ; 1950
            53             68      24  C1 00562 75$:   ADDL3   #36, (R8), R3                         ; 1960
                           01  03  A3  91 00566        CMPB    3(R3), #1                             ; 1966
                                   0C  1A 0056A        BGTRU   76$                                   ;
                        63 020E0000 8F  D0 0056C       MOVL    #34471936, (R3)                       ;
                               04  A3  D4 00573        CLRL    4(R3)                                 ;
                                   0D  11 00576        BRB     77$                                   ;
                                   63  B5 00578 76$:   TSTW    (R3)                                  ;
                                   09  13 0057A        BEQL    77$                                   ;
                                   53  DD 0057C        PUSHL   R3                                    ;
                00000000G  00      01  FB 0057E        CALLS   #1, STR$FREE1_DX                      ;
                           50      68  D0 00585 77$:   MOVL    (R8), R0                              ; 1967
                       04  A3  20  A0  D0 00588        MOVL    32(R0), 4(R3)                         ;
            63       01E4  C2  20  A0  C3 0058D        SUBL3   32(R0), 484(R2), (R3)                 ; 1968
      63        1C  A0  10         00  ED 00594        CMPZV   #0, #16, 28(R0), (R3)                 ; 1969
                                   0F  1E 0059A        BGEQU   78$                                   ;
                                   01  DD 0059C        PUSHL   #1                                    ;
                         01061154  8F  DD 0059E        PUSHL   #17174868                             ;
                00000000G  00      02  FB 005A4        CALLS   #2, LIB$STOP                          ;
                        51 00000000G  00  D0 005AB 78$: MOVL   FILTER, R1                            ; 1974
                                   A3  13 005B2        BEQL    73$                                   ;
                           50      52  D0 005B4        MOVL    R2, R0                                ; 1981
                     01E0  C0      63  D0 005B7        MOVL    (R3), 480(R0)                         ;
                     01E4  C2  04  A3  D0 005BC        MOVL    4(R3), 484(R2)                        ;
```

```
                          63 020E0000      8F  D0 005C2          MOVL    #34471936, (R3)                         ; 1982
                                   04      A3  D4 005C9          CLRL    4(R3)                                   ;
                                           7E  D4 005CC          CLRL    -(SP)                                   ; 1992
                                           53  DD 005CE          PUSHL   R3                                      ;
                                           7E  D4 005D0          CLRL    -(SP)                                   ;
                                     01E0  C2  9F 005D2          PUSHAB  480(R2)                                 ;
                                     FA12  CF  9F 005D6          PUSHAB  P.AAI                                   ; 1991
                                     02D0  C2  9F 005DA          PUSHAB  720(R2)                                 ; 1990
                                       04  AC  9F 005DE          PUSHAB  SCB                                     ; 1987
                                   61      07  FB 005E1          CALLS   #7, (R1)                                ; 1992
                                     015D  31  005E4          BRW     105$                                    ;
                4A        11     A2      05  E1 005E7 79$:     BBC     #5, 17(R2), 82$                         ; 2008
                7E               68      24  C1 005EC          ADDL3   #36, (R8), -(SP)                        ; 2010
                                     014C  C2  9F 005F0          PUSHAB  332(R2)                                 ;
                                           52  DD 005F4          PUSHL   R2                                      ;
                         0000V     CF      03  FB 005F6          CALLS   #3, SEARCH_FOR_STRING                   ;
                                   38      50  E9 005FB          BLBC    R0, 82$                                 ;
                         0178      D2        00  B8 0E 005FE     INSQUE  @0(R8), @376(R2)                        ; 2017
                                   50        04  AC  D0 00604     MOVL    SCB, R0                                 ; 2018
                                         01AC  C0  D4 00608     CLRL    428(R0)                                 ;
                                   51    01EC  C0  9E 0060C     MOVAB   492(R0), R1                             ; 2019
                         0224      C0        61  D0 00611     MOVL    (R1), 548(R0)                           ;
                                   01    01C8  C0  D1 00616     CMPL    456(R0), #1                             ; 2024
                                               12  1A 0061B     BGTRU   80$                                     ;
                                   01    0194  C0  D1 0061D     CMPL    404(R0), #1                             ; 2025
                                               0B  1A 00622     BGTRU   80$                                     ;
                                   01          61  D1 00624     CMPL    (R1), #1                                ; 2026
                                               06  1B 00627     BLEQU   80$                                     ;
                                         0224  C0  D7 00629     DECL    548(R0)                                 ; 2028
                                               02  11 0062D     BRB     81$                                     ; 2024
                                               61  D6 0062F 80$:  INCL    (R1)                                    ; 2033
                         02A7      C0          12  90 00631 81$:  MOVB    #18, 679(R0)                            ; 2034
                                               77  11 00636 82$:  BRB     91$                                     ; 2035
                                   53          68  D0 00638 83$:  MOVL    (R8), R3                                ; 2044
                                     04        09  D0 0063B       MOVL    #9, FUNCTION                            ; 2045
                5         0204     C2        03  E0 0063F       BBS     #3, 516(R2), 84$                        ; 2055
                4           2C     A3        02  E1 00645       BBC     #2, 44(R3), 85$                         ; 2056
                            04     AE        0A  D0 0064A 84$:  MOVL    #10, FUNCTION                           ; 2058
                6           11     A2        06  E1 0064E 85$:  BBC     #6, 17(R2), 86$                         ; 2063
                            04     AE        0B  D0 00653       MOVL    #11, FUNCTION                           ; 2065
                                               04  11 00657     BRB     87$                                     ;
                                       0282  C2  D6 00659 86$:  INCL    642(R2)                                 ; 2067
                                   50 00000000G  00  D0 0065D 87$:  MOVL    OUTPUT, R0                              ; 2073
                                               7E  D4 00664     CLRL    -(SP)                                   ; 2077
                                         24    A3  9F 00666     PUSHAB  36(R3)                                  ;
                                         0C    AE  9F 00669     PUSHAB  FUNCTION                                ; 2075
                                       02D0    C2  9F 0066C     PUSHAB  720(R2)                                 ;
                                               58  DD 00670     PUSHL   R8                                      ; 2077
                                   60          05  FB 00672     CALLS   #5, (R0)                                ;
                                   64          50  D0 00675     MOVL    R0, (R4)                                ;
                                               68  D4 00678     CLRL    (R8)                                    ; 2083
                    00000000G      8F         64  D1 0067A     CMPL    (R4), #PSM$_PENDING                     ; 2088
                                               0A  12 00681     BNEQ    88$                                     ;
                5           2C     A3        01  E0 00683       BBS     #1, 44(R3), 96$                         ; 2095
                                   64          01  D0 00688     MOVL    #1, (R4)                                ; 2098
                                               48  11 0068B     BRB     95$                                     ; 2099
                         0178      D2        63  0E 0068D 88$:  INSQUE  (R3), @376(R2)                          ; 2106
```

```
                        50      04     AC  D0 00692          MOVL    SCB, R0                                 ; 2110
                        01    0220     C0  D1 00696          CMPL    544(R0), #1                             ;
                                       73  12 0069B          BNEQ    99$                                     ;
                6E      2C      A3         01 E0 0069D          BBS     #1, 44(R3), 99$                         ; 2111
                      02A7      C0         08 90 006A2          MOVB    #8, 679(R0)                             ; 2113
                                           67 11 006A7          BRB     99$                                     ; 1263
                        05                 64 E8 006A9 89$:     BLBS    (R4), 92$                               ; 2123
                        66                 04 90 006AC 90$:     MOVB    #4, (R6)                                ; 2126
                                           7C 11 006AF 91$:     BRB     102$                                    ; 2127
                        01    0144     C2  D1 006B1 92$:     CMPL    324(R2), #1                             ; 2145
                                       15  12 006B6          BNEQ    94$                                     ;
                              0144     C2  9F 006B8          PUSHAB  324(R2)                                 ; 2150
                              022C     C2  9F 006BC          PUSHAB  556(R2)                                 ; 2149
                00000000G     00         02 FB 006C0          CALLS   #2, SMB$SEND_TO_JOBCTL                  ; 2150
                        11    A2         08 88 006C7 93$:     BISB2   #8, 17(R2)                              ; 2151
                                           3C 11 006CB          BRB     98$                                     ; 2152
                        53      10     A2  9E 006CD 94$:     MOVAB   16(R2), R3                              ; 2159
                05              63         0D E1 006D1          BBC     #13, (R3), 97$                          ;
                                66         08 90 006D5 95$:     MOVB    #8, (R6)                                ; 2162
                                           6D 11 006D8 96$:     BRB     106$                                    ; 2163
                                6A         63 E8 006DA 97$:     BLBS    (R3), 106$                              ; 2169
                66              63         0C E0 006DD          BBS     #12, (R3), 106$                         ; 2170
                28      0124    C2         05 E1 006E1          BBC     #5, 292(R2), 100$                       ; 2177
                27              63         0E E0 006E7          BBS     #14, (R3), 100$                         ; 2178
          08    AE        54    A2         02 C9 006EB          BISL3   #2, 84(R2), DEVICE_STATUS               ; 2182
                                     08    AE  9F 006F1          PUSHAB  DEVICE_STATUS                           ; 2184
                                           7E  7C 006F4          CLRQ    -(SP)                                   ;
                                   F8F6    CF  9F 006F6          PUSHAB  P.AAJ                                   ; 2185
                                   022C    C2  9F 006FA          PUSHAB  556(R2)                                 ; 2184
                00000000G     00         05  FB 006FE          CALLS   #5, SMB$SEND_TO_JOBCTL                  ;
                        01    A3         08  88 00705          BISB2   #8, 1(R3)                               ; 2190
                              64 00000000G 8F  D0 00709 98$:     MOVL    #PSM$_PENDING, (R4)                     ; 2191
                                           52  11 00710 99$:     BRB     110$                                    ; 2192
                                     0218  C2  D5 00712 100$:    TSTL    536(R2)                                 ; 2195
                                           03  12 00716          BNEQ    101$                                    ;
                                         00C6  31 00718          BRW     120$                                    ;
          43    01B3      C2               03  E1 0071B 101$:    BBC     #3, 435(R2), 110$                       ; 2197
                0088      C2         01EC  C2  D1 00721          CMPL    492(R2), 184(R2)                        ; 2198
                                           3A  1B 00728          BLEQU   110$                                    ;
                                66         0E  90 0072A          MOVB    #14, (R6)                               ; 2200
                                           35  11 0072D 102$:    BRB     110$                                    ; 1263
                                           65  D5 0072F 103$:    TSTL    (SERVICE)                               ; 2208
                                           31  13 00731          BEQL    110$                                    ;
                                           7E  7C 00733          CLRQ    -(SP)                                   ; 2218
                                   F8BB    CF  9F 00735          PUSHAB  P.AAK                                   ; 2219
                                   02D0    C2  9F 00739 104$:    PUSHAB  720(R2)                                 ; 2218
                                     04    AC  9F 0073D          PUSHAB  SCB                                     ; 2215
                        00          B5        05 FB 00740          CALLS   #5, @0(SERVICE)                         ; 2218
                                64         50  D0 00744 105$:    MOVL    R0, (R4)                                ;
                                           1B  11 00747 106$:    BRB     110$                                    ; 1263
                                50         6B  9A 00749 107$:    MOVZBL  (R11), R0                               ; 2229
          00    021C      C2               50  E5 0074C          BBCC    R0, 540(R2), 108$                       ;
          03      10      A2               02  E4 00752 108$:    BBSC    #2, 16(R2), 109$                        ; 2235
                                         0080  31 00757          BRW     118$                                    ;
                                   02A5  C2  95 0075A 109$:    TSTB    677(R2)                                 ; 2237
                                           7A  13 0075E          BEQL    118$                                    ;
                        10      A2               04  88 00760          BISB2   #4, 16(R2)                      ; 2239
```

```
                                                 7E  11 00764 110$:   BRB     121$                                    ; 1263
                                                 53  7C 00766 111$:   CLRQ    BUSY_STREAMS                            ; 2243
                                           0218  C2  D4 00768         CLRL    536(R2)                                 ; 2256
                                     10    A2    0402  8F  AA 0076C         BICW2   #1026, 16(R2)                   ; 2258
                                           50    0144  C2  9E 00772         MOVAB   324(R2), R0                     ; 2265
                                           05          D0  D1 00777         CMPL    (R0), #5                        ;
                                                       05  13 0077A         BEQL    112$                            ;
                                           03          60  D1 0077C         CMPL    (R0), #3                        ; 2266
                                                       03  12 0077F         BNEQ    113$                            ;
                                           60          08  D0 00781 112$:   MOVL    #8, (R0)                        ; 2268
                                           028E        C2  9F 00784 113$:   PUSHAB  654(R2)                         ; 2279
                                             54        A2  9F 00788         PUSHAB  84(R2)                          ; 2278
                                                       7E  D4 0078B         CLRL    -(SP)                           ; 2279
                                             14        A2  9F 0078D         PUSHAB  20(R2)                          ; 2276
                                                       50  DD 00790         PUSHL   R0                              ; 2279
                                           022C        C2  9F 00792         PUSHAB  556(R2)                         ; 2274
                          00000000G        00            06  FB 00796         CALLS   #6, SMB$SEND_TO_JOBCTL          ; 2279
                                                       51  D4 0079D         CLRL    I                               ; 2284
                                           50 00000000G0041  D0 0079F 114$:   MOVL    PSM$GL_SCBVEC[I], R0            ; 2287
                                                       0D  13 007A7         BEQL    116$                            ; 2288
                                           02          0C  A0  E9 007A9         BLBC    12(R0), 115$                    ; 2291
                                                       54  D6 007AD         INCL    ACTIVE_STREAMS                  ; 2293
                                02         10    A0          01  E1 007AF 115$:   BBC     #1, 16(R0), 116$        ; 2294
                                                       53  D6 007B4         INCL    BUSY_STREAMS                    ; 2296
                                E5               51          1F  F3 007B6 116$:   AOBLEQ  #31, I, 114$            ; 2288
                                                       54  D5 007BA         TSTL    ACTIVE_STREAMS                  ; 2303
                                                       0D  12 007BC         BNEQ    117$                            ;
                                             10000001  8F  DD 007BE         PUSHL   #268435457                      ; 2305
                          00000000G        00            01  FB 007C4         CALLS   #1, SYS$EXIT                    ;
                                                       53  D5 007CB 117$:   TSTL    BUSY_STREAMS                    ; 2310
                                                       15  12 007CD         BNEQ    121$                            ;
                                         F825    CF    9F 007CF         PUSHAB  P.AAL                           ; 2312
                          00000000G        00            01  FB 007D3         CALLS   #1, SYS$PURGWS                  ;
                                                       08  11 007DA 118$:   BRB     121$                            ; 1263
                                06         11    A2          02  E1 007DC 119$:   BBC     #2, 17(R2), 122$        ; 2321
                                                 66          10  90 007E1 120$:   MOVB    #16, (R6)                       ; 2323
                                               F81E  31 007E4 121$:   BRW     1$                              ;
                                                     04 007E7 122$:   RET                                       ; 2333

; Routine Size:  2024 bytes,    Routine Base:  CODE + 00AC
```

```
;  1412          2334  1 %SBTTL 'COMPLETE_SERVICE - record completion for async. function'
;  1413          2335  1 ! Functional Description:
;  1414          2336  1 !               Records completion of an asynchronous service function
;  1415          2337  1 !               (one that was originally completed with PSM$_PENDING)
;  1416          2338  1 !               and records the completion status.
;  1417          2339  1 !
;  1418          2340  1 ! Formal Parameters:
;  1419          2341  1 !               SMB_CONTEXT : address of a SCB or an IOB
;  1420          2342  1 !               USER_STATUS : address of longword contain completion status
;  1421          2343  1 !
;  1422          2344  1 ! Implicit Inputs:
;  1423          2345  1 !               none
;  1424          2346  1 !
;  1425          2347  1 ! Implicit Outputs:
;  1426          2348  1 !               none
;  1427          2349  1 !
;  1428          2350  1 ! Returned Value:
;  1429          2351  1 !               SS$_NORMAL
;  1430          2352  1 !
;  1431          2353  1 ! Side Effects:
;  1432          2354  1 !               SCB updated with completions status and DISPATCH called
;  1433          2355  1 !               to resume processing
;  1434          2356  1 !--
;  1435          2357  1 GLOBAL ROUTINE PSM$REPORT (
;  1436          2358  1         SMB_CONTEXT     : REF $LONGWORD,        ! SCB or IOB address
;  1437          2359  1         USER_STATUS     : REF $LONGWORD         ! Completion status
;  1438          2360  1         ) =
;  1439          2361  2 BEGIN
;  1440          2362  2
;  1441          2363  2 ! Setup parameter referencing values
;  1442          2364  2 !
;  1443          2365  2 PARAMETER_INDEX_ (SMB_CONTEXT, USER_STATUS);
;  1444          2366  2
;  1445          2367  2 LOCAL
;  1446          2368  2         SCB             : REF $BBLOCK;
;  1447          2369  2
;  1448          2370  2 ! Pick up the context value
;  1449          2371  2 !
;  1450          2372  2 SCB = .SMB_CONTEXT[];
;  1451          2373  2
;  1452          2374  2
;  1453          2375  2 ! If the structure type -- if SCB then we have an SCB, else
;  1454          2376  2 ! we have an IOB.
;  1455          2377  2 !
;  1456          2378  2 IF .SCB[PSM$B_TYPE] EQL PSM$K_STRUCTURE_SCB
;  1457          2379  2 THEN
;  1458          2380  3     BEGIN
;  1459          2381  3     ! SCB -- we are completing an INPUT function.  If not currently
;  1460          2382  3     ! pending then something is wrong.
;  1461          2383  3     !
;  1462          2384  3     IF .SCB[PSM$L_SERVICE_STATUS] NEQ PSM$_PENDING THEN CODEERR_ ;
;  1463          2385  3
;  1464          2386  3     ! Pick up completion status, default is normal.
;  1465          2387  3     !
;  1466          2388  3     SCB[PSM$L_SERVICE_STATUS] = SS$_NORMAL;
;  1467          2389  3     IF PARAMETER_PRESENT_ (USER_STATUS)
;  1468          2390  3     THEN
```

```
;  1469          2391  3          SCB[PSM$L_SERVICE_STATUS] = .USER_STATUS[];
;  1470          2392  3
;  1471          2393  3      ! Call function dispatch to resume processing
;  1472          2394  3      !
;  1473          2395  3      PSM$FUNCTION_DISPATCH (.SCB);
;  1474          2396  3      END
;  1475          2397  3
;  1476          2398  2  ELSE
;  1477          2399  2
;  1478          2400  3      BEGIN
;  1479          2401  3      ! We have an IOB -- we are completing an asycn. output request
;  1480          2402  3      !
;  1481          2403  3      LOCAL IOB : REF $BBLOCK;
;  1482          2404  3      LOCAL OUTPUT_STATUS : INITIAL (SS$_NORMAL);
;  1483          2405  3
;  1484          2406  3      ! Locate the IOB, check its structure type, and locate the SCB
;  1485          2407  3      !
;  1486          2408  3      IOB = .SCB;
;  1487          2409  3      IF .IOB[IOB_B_TYPE] NEQ PSM$K_STRUCTURE_IOB THEN CODEERR_ ;
;  1488          2410  3      SCB = .IOB[IOB_A_CONTEXT];
;  1489          2411  3
;  1490          2412  3      ! Pick up the output completion status if specified -- default is normal
;  1491          2413  3      !
;  1492          2414  3      IF PARAMETER_PRESENT_ (USER_STATUS) THEN OUTPUT_STATUS = .USER_STATUS[];
;  1493          2415  3
;  1494          2416  3      ! If no errors ...
;  1495          2417  3      !
;  1496          2418  3      IF .OUTPUT_STATUS
;  1497          2419  3      THEN
;  1498          2420  4          BEGIN
;  1499          2421  4
;  1500          2422  4          ! Update accounting
;  1501          2423  4          !
;  1502          2424  4          INCREMENT_ (SCB[PSM$L_OUTPUT_QIOS]);
;  1503          2425  4
;  1504          2426  4          ! If we have a checkpoint associated with this output buffer or
;  1505          2427  4          ! if we are marked as stalled ...
;  1506          2428  4          !
;  1507          2429  4          IF .IOB[IOB_V_CHECKPOINT_PENDING]
;  1508          2430  4          OR .$BBLOCK[SCB[PSM$L_DEVICE_STATUS], SMBMSG$V_STALLED]
;  1509          2431  4          THEN
;  1510          2432  5              BEGIN
;  1511          2433  5
;  1512          2434  5              ! Then prepare to notify the job controller
;  1513          2435  5              !
;  1514          2436  5              LOCAL CHECKPOINT : INITIAL (0);
;  1515          2437  5              LOCAL CKP_DESC : VECTOR [2] PRESET ([0]=0);
;  1516          2438  5              LOCAL REQUEST_RESPONSE : INITIAL (SMBMSG$K_TASK_STATUS);
;  1517          2439  5
;  1518          2440  5              ! Output completion indicates we are no longer stalled
;  1519          2441  5              !
;  1520          2442  5              $BBLOCK [SCB[PSM$L_DEVICE_STATUS],SMBMSG$V_STALLED] = 0;
;  1521          2443  5
;  1522          2444  5              ! If we are also pausing then set the request response
;  1523          2445  5              ! to PAUSE_TASK.  By default it is TASK_STATUS which indicates
;  1524          2446  5              ! an asynchronous (unexpected) message to the job controller.
;  1525          2447  5              !
```

```
;  1526     244B  5              IF .IOB[IOB_V_PAUSE_PENDING]
;  1527     2449  5              THEN
;  1528     2450  5                  REQUEST_RESPONSE = .SCB[PSM$L_REQUEST_RESPONSE];
;  1529     2451  5
;  1530     2452  5
;  1531     2453  5              ! If a checkpoint is present setup a descriptor for it
;  1532     2454  5              !
;  1533     2455  5              IF .IOB[IOB_V_CHECKPOINT_PENDING]
;  1534     2456  5              THEN
;  1535     2457  6                  BEGIN
;  1536     2458  6                  CKP_DESC[0] = SMBMSG$S_CHECKPOINT_DATA;
;  1537     2459  6                  CKP_DESC[1] = IOB[IOB_T_CHECKPOINT_DATA];
;  1538     2460  6                  CHECKPOINT = CKP_DESC;
;  1539     2461  5                  END;
;  1540     2462  5
;  1541     2463  5              ! Notify the job controller of one or more:
;  1542     2464  5              !
;  1543     2465  5              !   - we are not stalled
;  1544     2466  5              !   - we have paused
;  1545     2467  5              !   - here is a checkpoint update
;  1546     2468  5              !
;  1547     2469  5              SMB$SEND_TO_JOBCTL (
;  1548     2470  5                  SCB[PSM$L_STREAM_INDEX],        ! - stream number
;  1549     2471  5                  REQUEST_RESPONSE,               ! - request response
;  1550     2472  5                  0,                              ! - no accounting
;  1551     2473  5                  .CHECKPOINT,                    ! - checkpoint or 0
;  1552     2474  5                  SCB[PSM$L_DEVICE_STATUS]        ! - device status
;  1553     2475  5                  );
;  1554     2476  4              END;
;  1555     2477  4          END
;  1556     2478  3      ELSE
;  1557     2479  3
;  1558     2480  3          ! Store any errors other than cancel or abort
;  1559     2481  3          !
;  1560     2482  3          IF .OUTPUT_STATUS EQL SS$_CANCEL OR .OUTPUT_STATUS EQL SS$_ABORT
;  1561     2483  3          THEN
;  1562     2484  3              1
;  1563     2485  3          ELSE
;  1564     2486  3              PSM$STORE_ERRORS (.SCB, PSM$_WRITEERR, 1, SCB[PSM$Q_DEVICE_NAME],
;  1565     2487  3                  .OUTPUT_STATUS);
;  1566     2488  3
;  1567     2489  3      ! If we are flushing the output stream (that is, suspending further
;  1568     2490  3      ! input/format operations until all pending output has been printed)
;  1569     2491  3      ! then update the service status in the SCB with the output status.
;  1570     2492  3      !
;  1571     2493  3      IF .IOB[IOB_V_FLUSH_PENDING]
;  1572     2494  3      THEN
;  1573     2495  3          SCB[PSM$L_SERVICE_STATUS] = .OUTPUT_STATUS;
;  1574     2496  3
;  1575     2497  3      ! Release the IOB
;  1576     2498  3      !
;  1577     2499  3      INSERT_TAIL_ (IOB[IOB_Q_QLINKS], SCB[PSM$Q_BUFFER_QUEUE]);
;  1578     2500  3
;  1579     2501  3      ! Call dispatch to resume processing
;  1580     2502  3      !
;  1581     2503  3      PSM$FUNCTION_DISPATCH (.SCB);
;  1582     2504  2      END;
```

```
;  1583          2505  2
;  1584          2506  2 SS$_NORMAL
;  1585          2507  2
;  1586          2508  1 END;


                                                003C 00000          .ENTRY  PSM$REPORT, Save R2,R3,R4,R5                    ; 2357
                                    55 00000000G   00   9E 00002    MOVAB   LIB$STOP, R5                                   ;
                                    5E             0C   C2 00009    SUBL2   #12, SP                                         ;
                                    52         04  BC   D0 0000C    MOVL    @SMB_CONTEXT, SCB                               ; 2372
                                    03         08  A2   91 00010    CMPB    8(SCB), #3                                      ; 2378
                                                   2D   12 00014    BNEQ    3$                                              ;
                                    53       0220  C2   9E 00016    MOVAB   544(SCB), R3                                    ; 2384
                                00000000G   8F     63   D1 0001B    CMPL    (R3), #PSM$_PENDING                             ;
                                                   0B   13 00022    BEQL    1$                                              ;
                                                   01   DD 00024    PUSHL   #1                                              ;
                                       01061154    8F   DD 00026    PUSHL   #17174868                                       ;
                                    65             02   FB 0002C    CALLS   #2, LIB$STOP                                    ;
                                    63             01   D0 0002F 1$: MOVL    #1, (R3)                                        ; 2388
                                    02             6C   91 00032    CMPB    (AP), #2                                        ; 2389
                                                   09   1F 00035    BLSSU   2$                                              ;
                                               08  AC   D5 00037    TSTL    8(AP)                                           ;
                                                   04   13 0003A    BEQL    2$                                              ;
                                    63         08  BC   D0 0003C    MOVL    @USER_STATUS, (R3)                              ; 2391
                                              00A8 31   00040 2$:   BRW     12$                                             ; 2395
                                    54             01   D0 00043 3$: MOVL    #1, OUTPUT_STATUS                               ; 2400
                                    53             52   D0 00046    MOVL    SCB, IOB                                        ; 2408
                                    02         08  A3   91 00049    CMPB    8(IOB), #2                                      ; 2409
                                                   0B   13 0004D    BEQL    4$                                              ;
                                                   01   DD 0004F    PUSHL   #1                                              ;
                                       01061154    8F   DD 00051    PUSHL   #17174868                                       ;
                                    65             02   FB 00057    CALLS   #2, LIB$STOP                                    ;
                                    52         14  A3   D0 0005A 4$: MOVL    20(IOB), SCB                                    ; 2410
                                    02             6C   91 0005E    CMPB    (AP), #2                                        ; 2414
                                                   09   1F 00061    BLSSU   5$                                              ;
                                               08  AC   D5 00063    TSTL    8(AP)                                           ;
                                                   04   13 00066    BEQL    5$                                              ;
                                    54         08  BC   D0 00068    MOVL    @USER_STATUS, OUTPUT_STATUS                     ;
                                    48             54   E9 0006C 5$: BLBC    OUTPUT_STATUS, 9$                               ; 2418
                                              01E8 C2   D6 0006F    INCL    488(SCB)                                        ; 2424
                                    05         2C  A3   E8 00073    BLBS    44(IOB), 6$                                     ; 2429
                                60  54         A2   04   E1 00077    BBC     #4, 84(SCB), 10$                                ; 2430
                                                   50   D4 0007C 6$: CLRL    CHECKPOINT                                      ; 2432
                                               04  AE   7C 0007E    CLRQ    CKP_DESC                                        ; 2437
                                    6E             09   D0 00081    MOVL    #9, REQUEST_RESPONSE                            ;
                                    54         A2       10   8A 00084    BICB2   #16, 84(SCB)                                    ; 2442
                                05  2C         A3       03   E1 00088    BBC     #3, 44(IOB), 7$                                 ; 2448
                                    6E       0144  C2   D0 0008D    MOVL    324(SCB), REQUEST_RESPONSE                      ; 2450
                                    0D         2C  A3   E9 00092 7$: BLBC    44(IOB), 8$                                     ; 2455
                                    04         AE       18   D0 00096    MOVL    #24, CKP_DESC                                   ; 2458
                                    08         AE   30  A3   9E 0009A    MOVAB   48(R3), CKP_DESC+4                              ; 2459
                                    50             04   AE   9E 0009F    MOVAB   CKP_DESC, CHECKPOINT                            ; 2460
                                                   54   A2   9F 000A3 8$: PUSHAB  84(SCB)                                      ; 2474
                                                   50   DD 000A6    PUSHL   CHECKPOINT                                      ;
```

```
                                        7E  D4 000A8          CLRL    -(SP)                                   ;
                                  0C    AE  9F 000AA          PUSHAB  REQUEST_RESPONSE                        ; 2470
                                022C    C2  9F 000AD          PUSHAB  556(SCB)                                ;
                    00000000G   00      05  FB 000B1          CALLS   #5, SMB$SEND_TO_JOBCTL                  ; 2474
                                        22  11 000B8          BRB     10$                                     ; 2418
                    00000830    8F      54  D1 000BA  9$:     CMPL    OUTPUT_STATUS, #2096                    ; 2482
                                        19  13 000C1          BEQL    10$                                     ;
                                2C      54  D1 000C3          CMPL    OUTPUT_STATUS, #44                      ;
                                        14  13 000C6          BEQL    10$                                     ;
                                        54  DD 000C8          PUSHL   OUTPUT_STATUS                           ; 2487
                                  4C    A2  9F 000CA          PUSHAB  76(SCB)                                 ; 2486
                                        01  DD 000CD          PUSHL   #1                                      ;
                            010610D2    8F  DD 000CF          PUSHL   #17174738                               ;
                                        52  DD 000D5          PUSHL   SCB                                     ;
                       0000V    CF      05  FB 000D7          CALLS   #5, PSM$STORE_ERRORS                    ;
              05         2C     A3      01  E1 000DC  10$:    BBC     #1, 44(IOB), 11$                        ; 2493
                       0220     C2      54  D0 000E1          MOVL    OUTPUT_STATUS, 544(SCB)                 ; 2495
                       0178     D2      63  0E 000E6  11$:    INSQUE  (IOB), @376(SCB)                        ; 2499
                                        52  DD 000EB  12$:    PUSHL   SCB                                     ; 2503
                       F726     CF      01  FB 000ED          CALLS   #1, PSM$FUNCTION_DISPATCH               ;
                                50      01  D0 000F2          MOVL    #1, R0                                  ; 2508
                                            04 000F5          RET                                             ;

; Routine Size:  246 bytes,    Routine Base:  CODE + 0894
```

```
;  1588          2509  1 %SBTTL 'INCLUDE_MODULES - queue text modules for inclusion'
;  1589          2510  1 ! Functional Description:
;  1590          2511  1 !                   Adds the specified modules to the queue of modules
;  1591          2512  1 !                   that are waiting to be included in the input stream
;  1592          2513  1 !
;  1593          2514  1 ! Formal Parameters:
;  1594          2515  1 !                   SMB_CONTEXT     : assumed to be the SCB address
;  1595          2516  1 !                   MODULE_LIST     : descriptor of comma separate module list
;  1596          2517  1 !
;  1597          2518  1 ! Implicit Inputs:
;  1598          2519  1 !                   none
;  1599          2520  1 !
;  1600          2521  1 ! Implicit Outputs:
;  1601          2522  1 !                   none
;  1602          2523  1 !
;  1603          2524  1 ! Returned Value:
;  1604          2525  1 !                   none
;  1605          2526  1 !
;  1606          2527  1 ! Side Effects:
;  1607          2528  1 !                   The modules are appended to the module list
;  1608          2529  1 !--
;  1609          2530  1 GLOBAL ROUTINE PSM$INCLUDE_MODULES (
;  1610          2531  1         SMB_CONTEXT     : REF $LONGWORD,        ! SCB address
;  1611          2532  1         MODULE_LIST     : REF VECTOR            ! Module list descriptor
;  1612          2533  1         ) =
;  1613          2534  2 BEGIN
;  1614          2535  2
;  1615          2536  2 LOCAL SCB : REF $BBLOCK;
;  1616          2537  2
;  1617          2538  2
;  1618          2539  2 ! Locate the SCB
;  1619          2540  2 !
;  1620          2541  2 SCB = .SMB_CONTEXT[];
;  1621          2542  2
;  1622          2543  2
;  1623          2544  2 ! Check for empty list
;  1624          2545  2 !
;  1625          2546  2 IF .DESC_SIZE_ (.MODULE_LIST) EQL 0 THEN RETURN SS$_NORMAL;
;  1626          2547  2
;  1627          2548  2
;  1628          2549  2 ! If the pending list is non-empty then append a comma prior
;  1629          2550  2 ! to new modules
;  1630          2551  2 !
;  1631          2552  2 IF .DESC_SIZE_ (SCB[PSM$Q_MODULE_LIST]) NEQ 0
;  1632          2553  2 THEN
;  1633          2554  2     STR$APPEND (SCB[PSM$Q_MODULE_LIST], %ASCID ',');
;  1634          2555  2
;  1635          2556  2
;  1636          2557  2 ! Append the new modules
;  1637          2558  2 !
;  1638          2559  2 STR$APPEND (SCB[PSM$Q_MODULE_LIST], .MODULE_LIST);
;  1639          2560  2
;  1640          2561  2 SS$_NORMAL
;  1641          2562  2
;  1642          2563  1 END;
```

```
                                      0098A           .BLKB    2
                  00  00  00  2C      0098C P.AAN:    .ASCII   \,\<0><0><0>                       ;
                       010E0001       00990 P.AAM:    .LONG    17694721                          ;
                       00000000'      00994           .ADDRESS P.AAN                             ;


                              000C    00000           .ENTRY   PSM$INCLUDE_MODULES, Save R2,R3   ; 2530
           53 00000000G   00   9E     00002           MOVAB    STR$APPEND, R3                    ;
           50          04 BC   D0     00009           MOVL     @SMB_CONTEXT, SCB                 ; 2541
                       08 BC   B5     0000D           TSTW     @MODULE_LIST                      ; 2546
                          19   13     00010           BEQL     2$                                ;
           52        01CC C0   9E     00012           MOVAB    460(SCB), R2                      ; 2552
                          62   B5     00017           TSTW     (R2)                              ;
                          08   13     00019           BEQL     1$                                ;
                       DA AF   9F     0001B           PUSHAB   P.AAM                             ; 2554
                          52   DD     0001E           PUSHL    R2                                ;
           63             02   FB     00020           CALLS    #2, STR$APPEND                    ;
                       08 AC   DD     00023 1$:       PUSHL    MODULE_LIST                       ; 2559
                          52   DD     00026           PUSHL    R2                                ;
           63             02   FB     00028           CALLS    #2, STR$APPEND                    ;
           50             01   D0     0002B 2$:       MOVL     #1, R0                            ; 2563
                               04     0002E           RET                                        ;

; Routine Size:  47 bytes,    Routine Base:  CODE + 0998
```

```
;  1644     2564  1 %SBTTL 'PRINT_SYMBIONT - initialization/main entry point for print symbiont'
;  1645     2565  1 ! Functional Description:
;  1646     2566  1 !                   Initializes the print symbiont and begins processing
;  1647     2567  1 !
;  1648     2568  1 ! Formal Parameters:
;  1649     2569  1 !                   STREAMS :       Number of streams to allow (1-16)
;  1650     2570  1 !                   BUFLIM  :       Maximum output buffer size to allow
;  1651     2571  1 !                   USER_SIZE :     User work area size to allocate
;  1652     2572  1 !
;  1653     2573  1 ! Implicit Inputs:
;  1654     2574  1 !                   none
;  1655     2575  1 !
;  1656     2576  1 ! Implicit Outputs:
;  1657     2577  1 !                   none
;  1658     2578  1 !
;  1659     2579  1 ! Returned Value:
;  1660     2580  1 !                   none
;  1661     2581  1 !
;  1662     2582  1 ! Side Effects:
;  1663     2583  1 !                   Symbiont processing is initiated
;  1664     2584  1 !--
;  1665     2585  1 GLOBAL ROUTINE PSM$PRINT (
;  1666     2586  1         STREAMS         : REF $LONGWORD,
;  1667     2587  1         BUFLIM          : REF $WORD,
;  1668     2588  1         USER_SIZE       : REF $WORD
;  1669     2589  1         ) =
;  1670     2590  2 BEGIN
;  1671     2591  2
;  1672     2592  2 ! Setup for parameter referencing
;  1673     2593  2 !
;  1674     2594  2 PARAMETER_INDEX_ (STREAMS, BUFLIM, USER_SIZE);
;  1675     2595  2
;  1676     2596  2 BUILTIN FP;
;  1677     2597  2
;  1678     2598  2 LOCAL
;  1679     2599  2
;  1680     2600  2         ARG_DESC : $DYNAMIC_DESC,
;  1681     2601  2
;  1682     2602  2         ! Privileges needed by standard symbiont
;  1683     2603  2         !
;  1684     2604  2         PRIVILEGE_MASK: $BBLOCK[8] PRESET (
;  1685     2605  2             [PRV$V_ALLSPOOL]    = 1,
;  1686     2606  2             [PRV$V_LOG_IO]      = 1,
;  1687     2607  2             [PRV$V_PHY_IO]      = 1,
;  1688     2608  2             [PRV$V_READALL]     = 1,
;  1689     2609  2             [PRV$V_SHARE]       = 1),
;  1690     2610  2
;  1691     2611  2         MAXSTREAMS      : INITIAL (1)
;  1692     2612  2         ;
;  1693     2613  2
;  1694     2614  2 ! Create an item list for GETSYI call
;  1695     2615  2 !
;  1696     2616  2 BIND ITMLST = $ITMLST_UPLIT ((ITMCOD=SYI$_MAXBUF, BUFADR=PSM$GL_MAXBUF));
;  1697     2617  2
;  1698     2618  2
;  1699     2619  2 ! Establish the main signal handler
;  1700     2620  2 !
```

```
;  1701          2621  2 .FP = HANDLER;
;  1702          2622  2
;  1703          2623  2
;  1704          2624  2 ! Get the needed priv's
;  1705          2625  2 !
;  1706          2626  2 SIGNAL_IF_ERROR_ ($SETPRV (ENBFLG=1, PRVADR=PRIVILEGE_MASK));
;  1707          2627  2
;  1708          2628  2
;  1709          2629  2 ! Get the value of the sysgen parameter for maximum buffer size
;  1710          2630  2 !
;  1711          2631  2 SIGNAL_IF_ERROR_ ($GETSYIW (ITMLST=ITMLST));
;  1712          2632  2
;  1713          2633  2
;  1714          2634  2 ! Compute the maximum allowed buffer size as the smaller of the
;  1715          2635  2 ! system limit and the user limit, less 100 to allow for $QIO overhead
;  1716          2636  2 !
;  1717          2637  2 PSM$GL_MAXBUF = .PSM$GL_MAXBUF - 100;
;  1718          2638  2 IF PARAMETER_PRESENT_ (BUFLIM)
;  1719          2639  2 THEN
;  1720          2640  2     IF .PSM$GL_MAXBUF GTRU .BUFLIM[]
;  1721          2641  2     THEN
;  1722          2642  2         PSM$GL_MAXBUF = .BUFLIM[];
;  1723          2643  2
;  1724          2644  2
;  1725          2645  2 ! Store the maximum streams value supplied by the user
;  1726          2646  2 !
;  1727          2647  2 IF PARAMETER_PRESENT_ (STREAMS)
;  1728          2648  2 THEN
;  1729          2649  2     MAXSTREAMS = .STREAMS[];
;  1730          2650  2
;  1731          2651  2
;  1732          2652  2 ! Store the user context area size requested by the user
;  1733          2653  2 !
;  1734          2654  2 IF PARAMETER_PRESENT_ (USER_SIZE)
;  1735          2655  2 THEN
;  1736          2656  2     PSM$GL_USER_CTX  = .USER_SIZE[];
;  1737          2657  2
;  1738          2658  2
;  1739          2659  2 ! Call the SMB$ facility to initialize symbiont environment and
;  1740          2660  2 ! message interface to the job controller
;  1741          2661  2 !
;  1742        P 2662  2 SIGNAL_IF_ERROR_ (SMB$INITIALIZE (
;  1743        P 2663  2     UPLIT (SMBMSG$K_STRUCTURE_LEVEL),
;  1744        P 2664  2     PSM$RECEIVE_MESSAGE_AST,
;  1745          2665  2     MAXSTREAMS));
;  1746          2666  2
;  1747          2667  2
;  1748          2668  2 ! Purge the working set
;  1749          2669  2 !
;  1750          2670  2 $PURGWS (INADR=UPLIT (0, %X '7FFFFFFF'));
;  1751          2671  2
;  1752          2672  2
;  1753          2673  2 ! Loop forever at non-ast level, hibernating.  Nearly all symbiont activity
;  1754          2674  2 ! occurs at ast-level, but a few functions occur at non-ast.  If woken from
;  1755          2675  2 ! hibernate then look for non-ast work to do.
;  1756          2676  2 !
;  1757          2677  2 WHILE 1
```

```
;  1758          2678  2 DO
;  1759          2679  3     BEGIN
;  1760          2680  3     PSM$WAIT_FOR_NON_AST (ARG_DESC);
;  1761          2681  4         BEGIN
;  1762          2682  4
;  1763          2683  4         ! Argument list pointed to by arg_desc is a longword array of
;  1764          2684  4         ! the following values:
;  1765          2685  4         !
;  1766          2686  4         !       [0]         = SCB
;  1767          2687  4         !       [1]         = AST routine to activate after user routine
;  1768          2688  4         !       [2]         = AST parameter for AST routine
;  1769          2689  4         !       [3]         = User level routine
;  1770          2690  4         !       [4]         = User level argument count
;  1771          2691  4         !       [5]:[end] = User level argument list
;  1772          2692  4         !
;  1773          2693  4         LOCAL SCB : REF $BBLOCK;
;  1774          2694  4         BIND ARG_VECTOR = .DESC_ADDR_ (ARG_DESC) : VECTOR;
;  1775          2695  4
;  1776          2696  4         SCB = .ARG_VECTOR [0];
;  1777          2697  4         SCB[PSM$L_NON_AST_STATUS] = CALLG (ARG_VECTOR[4], .ARG_VECTOR[3]);
;  1778          2698  4
;  1779          2699  4         IF .ARG_VECTOR[1] NEQ 0
;  1780          2700  4         THEN
;  1781        P 2701  4             SIGNAL_IF_ERROR_ ($DCLAST (ASTADR=.ARG_VECTOR[1],
;  1782          2702  4                 ASTPRM=.ARG_VECTOR[2]));
;  1783          2703  3         END;
;  1784          2704  2     END;
;  1785          2705  2
;  1786          2706  2 SS$_NORMAL
;  1787          2707  2
;  1788          2708  1 END;


                                        009C7                   .BLKB   1
                                  104F  0004  009C8 P.AAO:      .WORD   4, 4175                                         ;
                                    00000000G 009CC             .ADDRESS PSM$GL_MAXBUF                                  ;
                                    00000000  009D0             .LONG   0                                               ;
                                    00000000  009D4             .LONG   0                                               ;
                                    00000001  009D8 P.AAP:      .LONG   1                                               ;
                                7FFFFFFF      009DC P.AAQ:      .LONG   0, 2147483647                                   ;

                                                    ITMLST=             P.AAO
                                                    .EXTRN  SYS$SETPRV, SYS$GETSYIW
                                                    .EXTRN  SYS$DCLAST

                                        003C  00000             .ENTRY  PSM$PRINT, Save R2,R3,R4,R5                     ; 2585
                                55 00000000G  00    9E  00002   MOVAB   PSM$GL_MAXBUF, R5                               ;
                                54 00000000G  00    9E  00009   MOVAB   LIB$SIGNAL, R4                                  ;
                                5E            0C    C2  00010   SUBL2   #12, SP                                         ;
                          06    AE      020E  8F    B0  00013   MOVW    #526, ARG_DESC+2                                ; 2600
                                  08    AE    D4  00019   CLRL    ARG_DESC+4                                              ;
                             80400090   8F    DD  0001C   PUSHL   #-2143289200                                        ; 2609
                          04    AE            08    B0  00022   MOVW    #8, PRIVILEGE_MASK+4                            ;
                                  06    AE    D4  00026   CLRL    PRIVILEGE_MASK+6                                        ;
                                        01    DD  00029   PUSHL   #1                                                  ;
                                6D      0000V CF    9E  0002B   MOVAB   HANDLER, (FP)                                    ; 2621
```

```
                                         7E  7C 00030        CLRQ    -(SP)                                  ; 2626
                                 0C      AE  9F 00032        PUSHAB  PRIVILEGE_MASK                         ;
                                         01  DD 00035        PUSHL   #1                                     ;
                  00000000G  00          04  FB 00037        CALLS   #4, SYS$SETPRV                         ;
                             52          50  D0 0003E        MOVL    R0, STATUS                             ;
                             05          52  E8 00041        BLBS    STATUS, 1$                             ;
                                         52  DD 00044        PUSHL   STATUS                                 ;
                             64          01  FB 00046        CALLS   #1, LIB$SIGNAL                         ;
                                         7E  7C 00049 1$:    CLRQ    -(SP)                                  ; 2631
                                         7E  D4 0004B        CLRL    -(SP)                                  ;
                                 94      AF  9F 0004D        PUSHAB  ITMLST                                 ;
                                         7E  7C 00050        CLRQ    -(SP)                                  ;
                                         7E  D4 00052        CLRL    -(SP)                                  ;
                  00000000G  00          07  FB 00054        CALLS   #7, SYS$GETSYIW                        ;
                             52          50  D0 0005B        MOVL    R0, STATUS                             ;
                             05          52  E8 0005E        BLBS    STATUS, 2$                             ;
                                         52  DD 00061        PUSHL   STATUS                                 ;
                             64          01  FB 00063        CALLS   #1, LIB$SIGNAL                         ;
                             65 00000064 8F  C2 00066 2$:    SUBL2   #100, PSM$GL_MAXBUF                    ; 2637
                             02          6C  91 0006D        CMPB    (AP), #2                               ; 2638
                                         11  1F 00070        BLSSU   3$                                     ;
                                 08      AC  D5 00072        TSTL    8(AP)                                  ;
                                         0C  13 00075        BEQL    3$                                     ;
   65        08   BC         10          00  ED 00077        CMPZV   #0, #16, @BUFLIM, PSM$GL_MAXBUF        ; 2640
                                         04  1E 0007D        BGEQU   3$                                     ;
                             65  08      BC  3C 0007F        MOVZWL  @BUFLIM, PSM$GL_MAXBUF                 ; 2642
                                         6C  95 00083 3$:    TSTB    (AP)                                   ; 2647
                                         09  13 00085        BEQL    4$                                     ;
                                 04      AC  D5 00087        TSTL    4(AP)                                  ;
                                         04  13 0008A        BEQL    4$                                     ;
                             6E  04      BC  D0 0008C        MOVL    @STREAMS, MAXSTREAMS                   ; 2649
                             03          6C  91 00090 4$:    CMPB    (AP), #3                               ; 2654
                                         0D  1F 00093        BLSSU   5$                                     ;
                                 0C      AC  D5 00095        TSTL    12(AP)                                 ;
                                         08  13 00098        BEQL    5$                                     ;
                  00000000G  00  0C      BC  3C 0009A        MOVZWL  @USER_SIZE, PSM$GL_USER_CTX            ; 2656
                                         5E  DD 000A2 5$:    PUSHL   SP                                     ; 2665
                                00000000G 00 9F 000A4        PUSHAB  PSM$RECEIVE_MESSAGE_AST                ;
                                 FF46    CF  9F 000AA        PUSHAB  P.AAP                                  ;
                  00000000G  00          03  FB 000AE        CALLS   #3, SMB$INITIALIZE                     ;
                             52          50  D0 000B5        MOVL    R0, STATUS                             ;
                             05          52  E8 000B8        BLBS    STATUS, 6$                             ;
                                         52  DD 000BB        PUSHL   STATUS                                 ;
                             64          01  FB 000BD        CALLS   #1, LIB$SIGNAL                         ;
                                 FF34    CF  9F 000C0 6$:    PUSHAB  P.AAQ                                  ; 2670
                  00000000G  00          01  FB 000C4        CALLS   #1, SYS$PURGWS                         ;
                                 0C      AE  9F 000CB 7$:    PUSHAB  ARG_DESC                               ; 2680
                  00000000G  00          01  FB 000CE        CALLS   #1, PSM$WAIT_FOR_NON_AST               ;
                             52  10      AE  D0 000D5        MOVL    ARG_DESC+4, R2                         ; 2694
                             53          62  D0 000D9        MOVL    (R2), SCB                              ; 2696
                         0C  B2  10      A2  FA 000DC        CALLG   16(R2), @12(R2)                        ; 2697
                       01DC  C3          50  D0 000E1        MOVL    R0, 476(SCB)                           ;
                                 04      A2  D5 000E6        TSTL    4(R2)                                  ; 2699
                                         E0  13 000E9        BEQL    7$                                     ;
                                         7E  D4 000EB        CLRL    -(SP)                                  ; 2702
                             7E  04      A2  7D 000ED        MOVQ    4(R2), -(SP)                           ;
                  00000000G  00          03  FB 000F1        CALLS   #3, SYS$DCLAST                         ;
```

```
                                52              50  DD 000F8        MOVL    R0, STATUS                              ;
                                CD              52  E8 000FB        BLBS    STATUS, 7$                              ;
                                                52  DD 000FE        PUSHL   STATUS                                  ;
                                64              01  FB 00100        CALLS   #1, LIB$SIGNAL                          ;
                                                C6  11 00103        BRB     7$                                      ; 2677

; Routine Size:  261 bytes,    Routine Base:  CODE + 09E4
```

```
;  1790          2709  1 %SBTTL 'STORE_ERRORS - store errors reported by user in SCB'
;  1791          2710  1 !++
;  1792          2711  1 ! Functional Description:
;  1793          2712  1 !               Store the vector of condition codes in the call
;  1794          2713  1 !               in the SCB.
;  1795          2714  1 !
;  1796          2715  1 ! Formal Parameters:
;  1797          2716  1 !               SMB_CONTEXT     : assumed to be SCB address
;  1798          2717  1 !               <8(AP)>         : begining of condition list
;  1799          2718  1 !
;  1800          2719  1 ! Implicit Inputs:
;  1801          2720  1 !               none
;  1802          2721  1 !
;  1803          2722  1 ! Implicit Outputs:
;  1804          2723  1 !               Error conditions and associcated text are stored
;  1805          2724  1 !
;  1806          2725  1 ! Returned Value:
;  1807          2726  1 !               SS$_NORMAL
;  1808          2727  1 !
;  1809          2728  1 ! Side Effects:
;  1810          2729  1 !               none
;  1811          2730  1 !--
;  1812          2731  1 GLOBAL ROUTINE PSM$STORE_ERRORS (
;  1813          2732  1         SMB_CONTEXT     : REF $LONGWORD
;  1814          2733  1         ) =
;  1815          2734  2 BEGIN
;  1816          2735  2
;  1817          2736  2 BUILTIN AP;
;  1818          2737  2 MAP     AP : REF VECTOR;
;  1819          2738  2
;  1820          2739  2 LOCAL
;  1821          2740  2         CONDITION,
;  1822          2741  2         ERRORS          : REF VECTOR,
;  1823          2742  2         INDEX           : INITIAL (0),
;  1824          2743  2         SCB             : REF $BBLOCK
;  1825          2744  2         ;
;  1826          2745  2
;  1827          2746  2 ! Locate the SCB and condition vector area
;  1828          2747  2 !
;  1829          2748  2 SCB = .SMB_CONTEXT[];
;  1830          2749  2 ERRORS = SCB[PSM$T_CONDITION_AREA];
;  1831          2750  2
;  1832          2751  2 ! If previous errors reported then ignore these
;  1833          2752  2 !
;  1834          2753  2 IF .ERRORS[0] NEQ 0 THEN RETURN SS$_NORMAL;
;  1835          2754  2
;  1836          2755  2
;  1837          2756  2 ! Expand the condition codes into a text message
;  1838          2757  2 !
;  1839          2758  2 EXPAND_CONDITION_VECTOR (.SCB, .AP[0] - 1, AP[2], SCB[PSM$Q_CONDITION_TEXT]);
;  1840          2759  2
;  1841          2760  2
;  1842          2761  2 ! Mark errrors to print
;  1843          2762  2 !
;  1844          2763  2 SERVICE_LIST_ (FILE_ERRORS) = 1;
;  1845          2764  2
;  1846          2765  2
```

```
;  1847          2766  2 ! Store the errors passing over FAO arguments
;  1848          2767  2 !
;  1849          2768  2 INCR I FROM 2 TO .AP[0]
;  1850          2769  2 DO
;  1851          2770  3     BEGIN
;  1852          2771  3     CONDITION = .AP [.I];
;  1853          2772  3     IF .CONDITION NEQ 0
;  1854          2773  3     THEN
;  1855          2774  4         BEGIN
;  1856          2775  4         INCREMENT_ (INDEX);
;  1857          2776  4         IF .INDEX GTRU PSM$S_CONDITION_AREA / 4  - 1
;  1858          2777  4         THEN
;  1859          2778  4             EXITLOOP;
;  1860          2779  4         INCREMENT_ (ERRORS[0]);
;  1861          2780  4         ERRORS[.INDEX] = .CONDITION;
;  1862          2781  3         END;
;  1863          2782  3
;  1864          2783  3     ! If this is neither an RMS nor a system message then
;  1865          2784  3     ! the low 16 bits of the next argument are an FAO count.
;  1866          2785  3     ! Skip the count argument longword, and the number of
;  1867          2786  3     ! additional longwords specified by the count.
;  1868          2787  3     !
;  1869          2788  3     IF .$BBLOCK [CONDITION,STS$V_FAC_NO] NEQ RMS$_FACILITY
;  1870          2789  3     AND .$BBLOCK [CONDITION,STS$V_FAC_NO] NEQ 0
;  1871          2790  3     AND .I LSSU .AP[0]
;  1872          2791  3     THEN
;  1873          2792  3         I = .I + .(AP[.I+1])<0,16,0> + 1;
;  1874          2793  2     END;
;  1875          2794  2
;  1876          2795  2
;  1877          2796  2 ! Any error initiates a task abort
;  1878          2797  2 !
;  1879          2798  2 ABORT_TASK (.SCB);
;  1880          2799  2
;  1881          2800  2 SS$_NORMAL
;  1882          2801  2
;  1883          2802  1 END;


                                    003C 00000          .ENTRY  PSM$STORE_ERRORS, Save R2,R3,R4,R5          ; 2730
                                55  D4 00002          CLRL    INDEX                                       ; 2733
                           52    04  BC  D0 00004      MOVL    @SMB_CONTEXT, SCB                           ; 2747
                           53  028E  C2  9E 00008      MOVAB   654(R2), ERRORS                             ; 2748
                                63  D5 0000D          TSTL    (ERRORS)                                    ; 2753
                                59  12 0000F          BNEQ    5$                                          ;
                             0198  C2  9F 00011        PUSHAB  408(SCB)                                    ; 2758
                               08  AC  9F 00015        PUSHAB  8(AP)                                       ;
                7E              6C  01  C3 00018      SUBL3   #1, (AP), -(SP)                             ;
                                52  DD 0001C          PUSHL   SCB                                         ;
                      0000V     CF  04  FB 0001E      CALLS   #4, EXPAND_CONDITION_VECTOR                  ;
                      021A      C2  04  88 00023      BISB2   #4, 538(SCB)                                ; 2763
                                50  01  D0 00028      MOVL    #1, I                                       ; 2768
                                    32  11 0002B      BRB     3$                                          ;
                                54  6C40  D0 0002D 1$: MOVL    (AP)[I], CONDITION                          ; 2771
```

```
                                                 0D  13 00031          BEQL    2$                                          ; 2772
                                                 55  D6 00033          INCL    INDEX                                       ; 2775
                                              04 55  D1 00035          CMPL    INDEX, #4                                   ; 2776
                                                 29  1A 00038          BGTRU   4$                                          ;
                                                 63  D6 0003A          INCL    (ERRORS)                                    ; 2779
                                           6345  54  D0 0003C          MOVL    CONDITION, (ERRORS)[INDEX]                  ; 2780
                01            54           0C  10  ED 00040 2$:       CMPZV   #16, #12, CONDITION, #1                     ; 2788
                                                 18  13 00045          BEQL    3$                                          ;
                00            54           0C  10  ED 00047          CMPZV   #16, #12, CONDITION, #0                     ; 2789
                                                 11  13 0004C          BEQL    3$                                          ;
                                           6C    50  D1 0004E          CMPL    I, (AP)                                     ; 2790
                                                 0C  1E 00051          BGEQU   3$                                          ;
                                        04 AC40  DF  00053          PUSHAL  4(AP)[I]                                    ; 2792
                                           51    9E  3C 00057          MOVZWL  @(SP)+, R1                                  ;
                                           50 01 A140 9E  0005A       MOVAB   1(R1)[I], I                                 ;
                            CA             50    6C  F3 0005F 3$:     AOBLEQ  (AP), I, 1$                                 ; 2768
                                                 52  DD 00063 4$:     PUSHL   SCB                                         ; 2798
                                     0000V  CF    01  FB 00065          CALLS   #1, ABORT_TASK                              ;
                                           50    01  D0 0006A 5$:     MOVL    #1, R0                                      ; 2802
                                                 04  0006D            RET                                                 ;

; Routine Size:  110 bytes,    Routine Base:  CODE + 0AE9
```

```
;  1885          2803  1 %SBTTL 'ABORT_TASK - aborts the current task'
;  1886          2804  1 ! Functional Description:
;  1887          2805  1 !                   Causes the current task to be aborted by setting abort
;  1888          2806  1 !                   flags and cancelling unneeded input services.
;  1889          2807  1 !
;  1890          2808  1 ! Formal Parameters:
;  1891          2809  1 !                   SCB     : SCB address
;  1892          2810  1 !
;  1893          2811  1 ! Implicit Inputs:
;  1894          2812  1 !                   none
;  1895          2813  1 !
;  1896          2814  1 ! Implicit Outputs:
;  1897          2815  1 !                   none
;  1898          2816  1 !
;  1899          2817  1 ! Returned Value:
;  1900          2818  1 !                   none
;  1901          2819  1 !
;  1902          2820  1 ! Side Effects:
;  1903          2821  1 !                   The current task is cancelled.
;  1904          2822  1 !--
;  1905          2823  1 ROUTINE ABORT_TASK (
;  1906          2824  1         SCB       : REF $BBLOCK
;  1907          2825  1         )         : NOVALUE =
;  1908          2826  2 BEGIN
;  1909          2827  2
;  1910          2828  2
;  1911          2829  2 ! If the main input routine has been requested but not
;  1912          2830  2 ! yet called with open, and if the file is actually opened
;  1913          2831  2 ! as evidenced by FAB_VALID being set, then close the file
;  1914          2832  2 ! directly since the main path will not call with a CLOSE function
;  1915          2833  2 !
;  1916          2834  2 IF .SERVICE_LIST_ (MAIN_INPUT)
;  1917          2835  2 AND .SCB[PSM$V_FAB_VALID]
;  1918          2836  2 THEN
;  1919          2837  2     $CLOSE (FAB=.SCB[PSM$A_FAB]);
;  1920          2838  2
;  1921          2839  2
;  1922          2840  2 ! Cancel any pending main input (file printing) and file setup.
;  1923          2841  2 !
;  1924          2842  2 SERVICE_LIST_ (MAIN_INPUT) = 0;
;  1925          2843  2 SERVICE_LIST_ (FILE_SETUP) = 0;
;  1926          2844  2
;  1927          2845  2
;  1928          2846  2 ! Turn on file trailer, job trailer, and/or job reset if the job controller
;  1929          2847  2 ! indicated they should occur on a task abort
;  1930          2848  2 !
;  1931          2849  2 IF .SEPARATE_FLAG_ (FILE_TRAILER_ABORT) THEN SERVICE_LIST_ (FILE_TRAILER)  = 1;
;  1932          2850  2 IF .SEPARATE_FLAG_ (JOB_TRAILER_ABORT)  THEN SERVICE_LIST_ (JOB_TRAILER)   = 1;
;  1933          2851  2 IF .SEPARATE_FLAG_ (JOB_RESET_ABORT)    THEN SERVICE_LIST_ (JOB_RESET)     = 1;
;  1934          2852  2
;  1935          2853  2
;  1936          2854  2 ! Clear any pending input modules
;  1937          2855  2 !
;  1938          2856  2 CLEAR_STRING_ (SCB[PSM$Q_MODULE_LIST]);
;  1939          2857  2
;  1940          2858  2
;  1941          2859  2 ! Set the master EOF flag to force wind-down while poping the input
```

```
; 1942          2860  2 ! service routine stack
; 1943          2861  2 !
; 1944          2862  2 SCB[PSM$V_EOF] = 1;
; 1945          2863  2
; 1946          2864  1 END;


                                                                   .EXTRN   SYS$CLOSE

                                               000C 00000 ABORT_TASK:
                                                                   .WORD    Save R2,R3                          ; 2823
                          52        04   AC   D0 00002             MOVL     SCB, R2                             ; 2834
                          53      0218   C2   9E 00006             MOVAB    536(R2), R3
                          10        02   A3   E9 0000B             BLBC     2(R3), 1$
         08       10      A2             04   E1 0000F             BBC      #4, 16(R2), 1$                      ; 2835
                                  0248   C2   DD 00014             PUSHL    584(R2)                             ; 2837
            00000000G     00             01   FB 00018             CALLS    #1, SYS$CLOSE
                   01     A3      0110   8F   AA 0001F 1$:         BICW2    #272, 1(R3)                         ; 2842
                          50      0154   C2   9E 00025             MOVAB    340(R2), R0                         ; 2849
         04               60             03   E1 0002A             BBC      #3, (R0), 2$
                   02     A3             08   88 0002E             BISB2    #8, 2(R3)
         04               60             09   E1 00032 2$:         BBC      #9, (R0), 3$                        ; 2850
                   02     A3             20   88 00036             BISB2    #32, 2(R3)
                                         60   95 0003A 3$:         TSTB     (R0)                                ; 2851
                                         04   18 0003C             BGEQ     4$
                   02     A3             10   88 0003E             BISB2    #16, 2(R3)
                          01      01CF   C2   91 00042 4$:         CMPB     463(R2), #1                         ; 2856
                                         11   1A 00047             BGTRU    5$
                          50      01CC   C2   9E 00049             MOVAB    460(R2), R0
                          60  020E0000   8F   D0 0004E             MOVL     #34471936, (R0)
                                    04   A0   D4 00055             CLRL     4(R0)
                                         14   11 00058             BRB      6$
                          50             52   D0 0005A 5$:         MOVL     R2, R0
                                  01CC   C0   B5 0005D             TSTW     460(R0)
                                         0B   13 00061             BEQL     6$
                                  01CC   C2   9F 00063             PUSHAB   460(R2)
            00000000G     00             01   FB 00067             CALLS    #1, STR$FREE1_DX
                   10     A2             04   88 0006E 6$:         BISB2    #4, 16(R2)                          ; 2862
                                              04 00072             RET                                          ; 2864

; Routine Size:  115 bytes,    Routine Base:  CODE + 0B57
```

```
;  1948          2865  1 %SBTTL 'CARRIAGE_CONTROL - compute carriage control'
;  1949          2866  1 ! Functional Description:
;  1950          2867  1 !                   Computes carriage control for input records with the
;  1951          2868  1 !                   assistance of the EXEC's carriage control routine.
;  1952          2869  1 !
;  1953          2870  1 ! Formal Parameters:
;  1954          2871  1 !                   SCB     : SCB address
;  1955          2872  1 !
;  1956          2873  1 ! Implicit Inputs:
;  1957          2874  1 !                   Carriage control type, first byte of input record,
;  1958          2875  1 !                   record header, form feed flags
;  1959          2876  1 !
;  1960          2877  1 ! Implicit Outputs:
;  1961          2878  1 !                   PSM$L_CARCON established
;  1962          2879  1 !
;  1963          2880  1 ! Returned Value:
;  1964          2881  1 !                   none
;  1965          2882  1 !
;  1966          2883  1 ! Side Effects:
;  1967          2884  1 !                   none
;  1968          2885  1 !--
;  1969          2886  1 ROUTINE CARRIAGE_CONTROL (
;  1970          2887  1         SCB     : REF $BBLOCK
;  1971          2888  1         ) =
;  1972          2889  2 BEGIN
;  1973          2890  2
;  1974          2891  2 ! Define JSB linkage to EXEC routine
;  1975          2892  2 !
;  1976          2893  2 LINKAGE
;  1977          2894  2         CARRIAGE_LINKAGE = JSB (REGISTER=3):
;  1978          2895  2                                 PRESERVE (3)
;  1979          2896  2                                 NOTUSED (2,4,5,6,7,8,9,10,11);
;  1980          2897  2
;  1981          2898  2 EXTERNAL ROUTINE
;  1982          2899  2         EXE$CARRIAGE:   CARRIAGE_LINKAGE NOVALUE;
;  1983          2900  2
;  1984          2901  2
;  1985          2902  2 ! Case on the carriage control type for this input routine
;  1986          2903  2 !
;  1987          2904  2 CASE .SCB[PSM$B_CC_TYPE] FROM 1 TO PSM$K_CC_MAX - 1 OF
;  1988          2905  2
;  1989          2906  2     SET
;  1990          2907  2
;  1991          2908  2     [OUTRANGE]:
;  1992          2909  2         CODEERR_ ;
;  1993          2910  2
;  1994          2911  2
;  1995          2912  2     ! Internal -- all carriage control is explicity imbedded in
;  1996          2913  2     ! the data records
;  1997          2914  2     !
;  1998          2915  2
;  1999          2916  2     [PSM$K_CC_INTERNAL]:
;  2000          2917  2         SCB[PSM$L_CARCON] = 0;
;  2001          2918  2
;  2002          2919  2
;  2003          2920  2
;  2004          2921  2     ! Implied -- generate leading <CR> and trailing <LF> for most
```

```
;2005     922  2      ! records with special handling for the first record from the
;2006     923  2      ! service and for form feeds in the first byte of a record.
;2007     924  2      !
;2008     925  2
;2009    2926  2      [PSM$K_CC_IMPLIED]:
;2010    2927  3          BEGIN
;2011    2928  3
;2012    2929  3          ! Default carriage control
;2013    2930  3          !
;2014    2931  3          SCB[PSM$L_CARCON] = PSM$K_LF_CR;
;2015    2932  3
;2016    2933  3
;2017    2934  3          ! Clear leading carriage control for first record from service
;2018    2935  3          !
;2019    2936  3          IF .SCB[PSM$V_FIRST_RECORD]
;2020    2937  3          THEN
;2021    2938  3              SCB[PSM$B_PREFIX_COUNT] = 0;
;2022    2939  3
;2023    2940  3
;2024    2941  3          ! Clear leading carriage control if last record was FF only
;2025    2942  3          !
;2026    2943  3          IF TESTBITSC (SCB[PSM$V_IMPLICIT_FORMFEED])
;2027    2944  3          THEN
;2028    2945  3              SCB[PSM$B_PREFIX_COUNT] = 0;
;2029    2946  3
;2030    2947  3
;2031    2948  3          ! Check for form feed in first byte of record
;2032    2949  3          !
;2033    2950  3          IF .SCB_SIZE_ (INPUT_RECORD) GTRU 0
;2034    2951  3          THEN
;2035    2952  3              IF CH$RCHAR (.SCB_ADDR_ (INPUT_RECORD)) EQL PSM$K_CHAR_FF
;2036    2953  3              THEN
;2037    2954  4                  BEGIN
;2038    2955  4
;2039    2956  4                  ! First byte is form feed -- clear leading carriage control
;2040    2957  4                  !
;2041    2958  4                  SCB[PSM$B_PREFIX_COUNT] = 0;
;2042    2959  4
;2043    2960  4                  ! One byte record -- clear trailing carriage control and set
;2044    2961  4                  ! implicit form feed flag to clear leading carriage control
;2045    2962  4                  ! for next record
;2046    2963  4                  !
;2047    2964  4                  IF .SCB_SIZE_ (INPUT_RECORD) EQL 1
;2048    2965  4                  THEN
;2049    2966  5                      BEGIN
;2050    2967  5                      SCB[PSM$B_POSTFIX_COUNT] = 0;
;2051    2968  5                      SCB[PSM$V_IMPLICIT_FORMFEED] = 1;
;2052    2969  4                      END;
;2053    2970  3                  END;
;2054    2971  2          END;
;2055    2972  2
;2056    2973  2
;2057    2974  2      ! Fortran -- first byte of the record defines carriage control
;2058    2975  2      !
;2059    2976  2
;2060    2977  2      [PSM$K_CC_FORTRAN]:
;2061    2978  2          IF .SCB_SIZE_ (INPUT_RECORD) EQL 0
```

```
2               THEN
2                   SCB[PSM$L_CARCON] = PSM$K_LF_CR
2               ELSE
3                   BEGIN
3                   SCB[PSM$L_CARCON] = CH$RCHAR (.SCB_ADDR (INPUT_RECORD));
3                   EXE$CARRIAGE (SCB[PSM$L_CAR:ON] - $BYTEOFFSET (IRP$B_CARCON));
3                   IF .SCB[PSM$B_PREFIX_CHAR] EuL 0
3                   THEN
3                       SCB[PSM$B_PREFIX_CHAR] = PSM$K_CHAR_LF;
3                   IF .SCB[PSM$B_POSTFIX_CHAR] EQL 0
3                   THEN
3                       SCB[PSM$B_POSTFIX_CHAR] = PSM$K_CHAR_LF;
3                   RETURN PSM$K_FIRST_CHAR_USED;
2                   END;
2
2
2
2           ! PRINT -- print file format (PRN).  Each record has a two byte
2           ! header that define carriage control.  DCL, for example, creates
2           ! PRN files.
2           !
2
2           [PSM$K_CC_PRINT]:
3               BEGIN
3               SCB[PSM$L_CARCON] = .SCB[PSM$L_RECORD_HEADER] ^ 16;
3               EXE$CARRIAGE (SCB[PSM$L_CARCON] - $BYTEOFFSET (IRP$B_CARCON));
3               IF .SCB[PSM$B_PREFIX_CHAR] EQL 0
3               THEN
3                   SCB[PSM$B_PREFIX_CHAR] = PSM$K_CHAR_LF;
3               IF .SCB[PSM$B_POSTFIX_CHAR] EQL 0
3               THEN
3                   SCB[PSM$B_POSTFIX_CHAR] = PSM$K_CHAR_LF
2               END;
2
2           TES;
2
2   RETURN SS$_NORMAL;
2
1   END;


                                                          .EXTRN  EXE$CARRIAGE

                                001C 00000 CARRIAGE_CONTROL:
                                                          .WORD   Save R2,R3,R4                  ; 2886
             54 00000000G   00   9E 00002                 MOVAB   EXE$CARRIAGE, R4
             52        04   AC   D0 00009                 MOVL    SCB, R2                         ; 2904
      03     01      027C   C2   8F 0000D                 CASEB   636(R2), #1, #3
0095  005A         001F        0019    00013 1$:          .WORD   2$-1$,-                         ;
                                                                  3$-1$,-                         ;
                                                                  6$-1$,-                         ;
                                                                  11$-1$                          ;
                              01   DD 0001B                 PUSHL   #1                              ; 2908
                 0106:154     8F   DD 0001D                 PUSHL   #17174868                       ;
      00000000G  00           02   FB 00023                 CALLS   #2, LIB$STOP                    ;
                              56   11 0002A                 BRB     7$                              ;
```

```
                              0278  C2  D4 0002C 2$:     CLRL    632(R2)                         ; 2917
                                    50  11 00030         BRB     7$                              ;
                        51    0278  C2  9E 00032 3$:     MOVAB   632(R2), R1                     ; 2931
                        61 0D010A01 8F  D0 00037         MOVL    #218171905, (R1)                ;
        02        10    A2          05  E1 0003E         BBC     #5, 16(R2), 4$                  ; 2936
                                    61  94 00043         CLRB    (R1)                            ; 2938
        02        10    A2          06  E5 00045 4$:     BBCC    #6, 16(R2), 5$                  ; 2943
                                    61  94 0004A         CLRB    (R1)                            ; 2945
                        50    0260  C2  9E 0004C 5$:     MOVAB   608(R2), R0                     ; 2950
                                    60  B5 00051         TSTW    (R0)                            ;
                                    78  13 00053         BEQL    13$                             ;
                        0C      04  B0  91 00055         CMPB    @4(R0), #12                     ; 2952
                                    72  12 00059         BNEQ    13$                             ;
                                    61  94 0005B         CLRB    (R1)                            ; 2958
                        01          60  B1 0005D         CMPW    (R0), #1                        ; 2964
                                    6B  12 00060         BNEQ    13$                             ;
                              027A  C2  94 00062         CLRB    634(R2)                         ; 2967
                  10    A2      40  8F  88 00066         BISB2   #64, 16(R2)                     ; 2968
                                    60  11 0006B         BRB     13$                             ; 2904
                        51    0278  C2  9E 0006D 6$:     MOVAB   632(R2), R1                     ; 2980
                        50    0260  C2  9E 00072         MOVAB   608(R2), R0                     ; 2978
                                    60  B5 00077         TSTW    (R0)                            ;
                                    09  12 00079         BNEQ    8$                              ;
                        61 0D010A01 8F  D0 0007B         MOVL    #218171905, (R1)                ; 2980
                                    49  11 00082 7$:     BRB     13$                             ;
                        61      04  B0  9A 00084 8$:     MOVZBL  @4(R0), (R1)                    ; 2983
                        53      C4  A1  9E 00088         MOVAB   -60(R1), R3                     ; 2984
                                    64  16 0008C         JSB     EXE$CARRIAGE                    ;
                              0279  C2  95 0008E         TSTB    633(R2)                         ; 2985
                                    05  12 00092         BNEQ    9$                              ;
                0279    C2          0A  90 00094         MOVB    #10, 633(R2)                    ; 2987
                              027B  C2  95 00099 9$:     TSTB    635(R2)                         ; 2988
                                    05  12 0009D         BNEQ    10$                             ;
                027B    C2          0A  90 0009F         MOVB    #10, 635(R2)                    ; 2990
                        50          03  D0 000A4 10$:    MOVL    #3, R0                          ; 2991
                                        04 000A7         RET                                     ;
    027B    C2  0268    C2          10  78 000A8 11$:    ASHL    #16, 616(R2), 632(R2)           ; 3003
                        53    023C  C2  9E 000B0         MOVAB   572(R2), R3                     ; 3004
                                    64  16 000B5         JSB     EXE$CARRIAGE                    ;
                              0279  C2  95 000B7         TSTB    633(R2)                         ; 3005
                                    05  12 000BB         BNEQ    12$                             ;
                0279    C2          0A  90 000BD         MOVB    #10, 633(R2)                    ; 3007
                              027B  C2  95 000C2 12$:    TSTB    635(R2)                         ; 3008
                                    05  12 000C6         BNEQ    13$                             ;
                027B    C2          0A  90 000C8         MOVB    #10, 635(R2)                    ; 3010
                        50          01  D0 000CD 13$:    MOVL    #1, R0                          ; 3015
                                        04 000D0         RET                                     ; 3017

; Routine Size:  209 bytes,    Routine Base:  CODE + 0BCA
```

```
;  2102          3018  1 %SBTTL 'ENQUEUE_CHECKPOINT - add a checkpoint to the checkpoint queue'
;  2103          3019  1 ! Functional Description:
;  2104          3020  1 !                   This routine manages additions to the checkpoint queue.
;  2105          3021  1 !
;  2106          3022  1 ! Formal Parameters:
;  2107          3023  1 !                   SCB     : SCB address
;  2108          3024  1 !                   CKP_DESC: address of the checkpoint descriptor
;  2109          3025  1 !
;  2110          3026  1 ! Implicit Inputs:
;  2111          3027  1 !                   Checkpoint queue header
;  2112          3028  1 !
;  2113          3029  1 ! Implicit Outputs:
;  2114          3030  1 !                   none
;  2115          3031  1 !
;  2116          3032  1 ! Returned Value:
;  2117          3033  1 !                   none
;  2118          3034  1 !
;  2119          3035  1 ! Side Effects:
;  2120          3036  1 !                   The checkpoint is enqueued.  Memory may be allocated.
;  2121          3037  1 !                   The queue may be flushed.
;  2122          3038  1 !--
;  2123          3039  1 ROUTINE ENQUEUE_CHECKPOINT (
;  2124          3040  1         SCB             : REF $BBLOCK,
;  2125          3041  1         CKP_DESC        : REF VECTOR
;  2126          3042  1         ) : NOVALUE =
;  2127          3043  2 BEGIN
;  2128          3044  2
;  2129          3045  2 LOCAL
;  2130          3046  2         DSB             : REF $BBLOCK
;  2131          3047  2         ;
;  2132          3048  2
;  2133          3049  2
;  2134          3050  2 ! If the queue has reached its maximum depth then flush it by
;  2135          3051  2 ! discarding every other checkpoint
;  2136          3052  2 !
;  2137          3053  2 IF .SCB[PSM$B_CHECKPOINT_DEPTH] GTR PSM$K_CHECKPOINT_LIMIT
;  2138          3054  2 THEN
;  2139          3055  3     BEGIN
;  2140          3056  3     LOCAL FIRST_DSB     : REF $BBLOCK,
;  2141          3057  3           TOGGLE        : INITIAL (0);
;  2142          3058  3
;  2143          3059  3
;  2144          3060  3     ! Scan the queue by removing each checkpoint.  Every other
;  2145          3061  3     ! checkpoint is requeued.
;  2146          3062  3     !
;  2147          3063  3     FIRST_DSB = .FLINK_ (SCB[PSM$Q_CHECKPOINT_QUEUE]);
;  2148          3064  3     DO
;  2149          3065  4         BEGIN
;  2150          3066  4         REMOVE_HEAD_ (DSB, SCB[PSM$Q_CHECKPOINT_QUEUE]);
;  2151          3067  4         DSB = .DSB - $BYTEOFFSET (DSB_Q_QLINKS);
;  2152          3068  4         IF .TOGGLE
;  2153          3069  4         THEN
;  2154          3070  5             BEGIN
;  2155          3071  5             PSM$DEALLOCATE_DSB (.DSB);
;  2156          3072  5             DECREMENT  (SCB[PSM$B_CHECKPOINT_DEPTH]);
;  2157          3073  5             IF .SCB[PSM$B_CHECKPOINT_DEPTH] LSS 0 THEN CODEERR_ ;
;  2158          3074  5             END
```

```
4             ELSE
4                 INSERT_TAIL_ (DSB[DSB_Q_QLINKS], SCB[PSM$Q_CHECKPOINT_QUEUE]);
4             INCREMENT_ (TOGGLE);
4             END
3         UNTIL
3             .FLINK_ (SCB[PSM$Q_CHECKPOINT_QUEUE]) EQL .FIRST_DSB;
2         END;
2
2
2 ! Allocate a dynamic string block, copy and enqueue the checkpoint
2 !
2 PSM$ALLOCATE_DSB (DSB);
2 COPY_DX_DX_ (CKP_DESC[0], DSB[DSB_Q_DESC]);
2 INSERT_TAIL_ (DSB[DSB_Q_QLINKS], SCB[PSM$Q_CHECKPOINT_QUEUE]);
2
2
2 ! Increment the checkpoint depth and check for coding error
2 !
2 INCREMENT_ (SCB[PSM$B_CHECKPOINT_DEPTH]);
2 IF .SCB[PSM$B_CHECKPOINT_DEPTH] LSS 0 THEN CODEERR_ ;   ! > 128
2
2 SS$_NORMAL
2
1 END;
```

```
                                      003C 00000 ENQUEUE_CHECKPOINT:
                                                         .WORD   Save R2,R3,R4,R5                    ; 3039
                      55 00000000G  00  9E 00002         MOVAB   LIB$STOP, R5                        ;
                      5E            04  C2 00009         SUBL2   #4, SP                              ;
                      50        04  AC  D0 0000C         MOVL    SCB, R0                             ; 3053
                      14      02A2  C0  91 00010         CMPB    674(R0), #20                        ;
                                54  15 00015         BLEQ    4$                                  ;
                                53  D4 00017         CLRL    TOGGLE                              ; 3055
                      54      017C  C0  D0 00019         MOVL    380(R0), FIRST_DSB                  ; 3063
       52        04   AC 0000017C  8F  C1 0001E         ADDL3   #380, SCB, R2                       ; 3066
                      6E        00  B2  0F 00027 1$:     REMQUE  @0(R2), DSB                         ;
                      23            53  E9 0002B         BLBC    TOGGLE, 2$                          ; 3068
                                6E  DD 0002E         PUSHL   DSB                                 ; 3071
           00000000G  00            01  FB 00030         CALLS   #1, PSM$DEALLOCATE_DSB              ;
                      50        04  AC  D0 00037         MOVL    SCB, R0                             ; 3072
                      50      02A2  C0  9E 0003B         MOVAB   674(R0), R0                         ;
                                60  97 00040         DECB    (R0)                                ;
                                17  18 00042         BGEQ    3$                                  ; 3073
                                01  DD 00044         PUSHL   #1                                  ;
                        01061154    8F  DD 00046         PUSHL   #17174868                           ;
                      65            02  FB 0004C         CALLS   #2, LIB$STOP                        ;
                                0A  11 0004F         BRB     3$                                  ; 3068
                      50        04  AC  D0 00051 2$:     MOVL    SCB, R0                             ; 3076
                0180  D0        00  BE  0E 00055         INSQUE  @DSB, @384(R0)                      ;
                                53  D6 0005B 3$:     INCL    TOGGLE                              ; 3077
       52        04   AC 0000017C  8F  C1 0005D         ADDL3   #380, SCB, R2                       ; 3080
                      54            62  D1 00066         CMPL    (R2), FIRST_DSB                     ;
                                BC  12 00069         BNEQ    1$                                  ;
```

```
                                      5E  DD 0006B 4$:     PUSHL   SP                                   ; 3086
            00000000G  00             01  FB 0006D         CALLS   #1, PSM$ALLOCATE_DSB                 ;
                             08       AC  DD 00074         PUSHL   CKP_DESC                             ; 3087
      7E          04   AE             08  C1 00077         ADDL3   #8, DSB, -(SP)                       ;
            00000000G  00             02  FB 0007C         CALLS   #2, STR$COPY_DX                      ;
                       52             50  D0 00083         MOVL    R0, STATUS                           ;
                       09             52  E8 00086         BLBS    STATUS, 5$                           ;
                                      52  DD 00089         PUSHL   STATUS                               ;
            00000000G  00             01  FB 0008B         CALLS   #1, LIB$SIGNAL                       ;
                       50        04   AC  D0 00092 5$:     MOVL    SCB, R0                              ; 3088
                 0180  D0        00   BE  0E 00096         INSQUE  @DSB, @384(R0)                       ;
                       50        04   AC  D0 0009C         MOVL    SCB, R0                              ; 3093
                       50      02A2   C0  9E 000A0         MOVAB   674(R0), R0                          ;
                                      60  96 000A5         INCB    (R0)                                 ;
                                      0B  18 000A7         BGEQ    6$                                   ; 3094
                                      01  DD 000A9         PUSHL   #1                                   ;
                         01061154     8F  DD 000AB         PUSHL   #17174868                            ;
                       65             02  FB 000B1         CALLS   #2, LIB$STOP                         ;
                                          04 000B4 6$:     RET                                          ; 3098

; Routine Size:  181 bytes,    Routine Base:  CODE + 0C9B
```

```
;  2184          3099  1 %SBTTL 'EXPAND_CONDITION_VECTOR - expand condition codes to text'
;  2185          3100  1 ! Functional Description:
;  2186          3101  1 !                   Expands a list of condition codes to concatenated
;  2187          3102  1 !                   text messages.
;  2188          3103  1 !
;  2189          3104  1 ! Formal Parameters:
;  2190          3105  1 !                   SCB     : SCB address
;  2191          3106  1 !                   MSGCNT  : number of longwords in message vector
;  2192          3107  1 !                   MSGVEC  : address of message vector
;  2193          3108  1 !                   DESC    : address of descriptor to receive text
;  2194          3109  1 !
;  2195          3110  1 ! Implicit Inputs:
;  2196          3111  1 !                   none
;  2197          3112  1 !
;  2198          3113  1 ! Implicit Outputs:
;  2199          3114  1 !                   none
;  2200          3115  1 !
;  2201          3116  1 ! Returned Value:
;  2202          3117  1 !                   none
;  2203          3118  1 !
;  2204          3119  1 ! Side Effects:
;  2205          3120  1 !                   none
;  2206          3121  1 !--
;  2207          3122  1 GLOBAL ROUTINE EXPAND_CONDITION_VECTOR (
;  2208          3123  1         SCB     : REF $BBLOCK,
;  2209          3124  1         MSGCNT  ,
;  2210          3125  1         MSGVEC  : REF VECTOR,
;  2211          3126  1         DESC    : REF VECTOR            ! Dynamic descriptor to receive message
;  2212          3127  1         )       : NOVALUE =
;  2213          3128  2 BEGIN
;  2214          3129  2
;  2215          3130  2 BUILTIN AP;
;  2216          3131  2 LOCAL   TEMP : VECTOR [20];
;  2217          3132  2
;  2218          3133  2
;  2219          3134  2 ! Create a vector with message count in front, followed by messages
;  2220          3135  2 !
;  2221          3136  2 TEMP[0] = .MSGCNT;
;  2222          3137  2 CH$COPY (.MSGCNT * 4, .MSGVEC, 0, %ALLOCATION (TEMP) - 4, TEMP[1]);
;  2223          3138  2
;  2224          3139  2
;  2225          3140  2 ! Call $PUTMSG to look up text
;  2226          3141  2 !
;  2227        P 3142  2 SIGNAL_IF_ERROR_ ($PUTMSG (MSGVEC=TEMP, ACTRTN=PUTMSG_ACTION,
;  2228          3143  2                   ACTPRM=.DESC));
;  2229          3144  2
;  2230          3145  1 END;


                                                                .EXTRN  SYS$PUTMSG

                                                003C 00000      .ENTRY  EXPAND_CONDITION_VECTOR, Save R2,R3,R4,R5       ; 3122
                                     5E     B4  AE  9E 00002    MOVAB   -76(SP), SP                                     ;
                                         08  AC  DD 00006    PUSHL   MSGCNT                                          ; 3136
                                 50      08  AC  02  78 00009    ASHL    #2, MSGCNT, R0                              ; 3137
004C    8F               00      0C  BC      50  2C 0000E    MOVC5   R0, @MSGVEC, #0, #76, TEMP+4                    ;
```

```
                                 04   AE      00016
                                 10   AC  DD 00018          PUSHL   DESC                                    ; 3143
                                      7E  D4 0001B          CLRL    -(SP)
                               0000V  CF  9F 0001D          PUSHAB  PUTMSG_ACTION
                                  0C  AE  9F 00021          PUSHAB  TEMP
                    00000000G  00         04  FB 00024     CALLS   #4, SYS$PUTMSG
                               52         50  D0 0002B     MOVL    R0, STATUS
                               09         52  E8 0002E     BLBS    STATUS, 1$
                                          52  DD 00031     PUSHL   STATUS
                    00000000G  00         01  FB 00033     CALLS   #1, LIB$SIGNAL
                                          04 0003A 1$:     RET                                             ; 3145

; Routine Size:  59 bytes,    Routine Base:  CODE + 0D50
```

```
;  2232          3146  1 %SBTTL 'FIND_CHECKPOINT -- locate an appropriate checkpoint'
;  2233          3147  1 ! Functional Description:
;  2234          3148  1 !                   Searches the checkpoint queue for the closest checkpoint
;  2235          3149  1 !                   that preceeds the target page.
;  2236          3150  1 !
;  2237          3151  1 ! Formal Parameters:
;  2238          3152  1 !                   SCB:    SCB ADDRESS
;  2239          3153  1 !
;  2240          3154  1 ! Implicit Inputs:
;  2241          3155  1 !                   Checkpoint queue, start page
;  2242          3156  1 !
;  2243          3157  1 ! Implicit Outputs:
;  2244          3158  1 !                   none
;  2245          3159  1 !
;  2246          3160  1 ! Returned Value:
;  2247          3161  1 !                   Address of checkpoint or zero
;  2248          3162  1 !
;  2249          3163  1 ! Side Effects:
;  2250          3164  1 !                   none
;  2251          3165  1 !--
;  2252          3166  1 ROUTINE FIND_CHECKPOINT (
;  2253          3167  1         SCB             : REF $BBLOCK
;  2254          3168  1         ) =
;  2255          3169  2 BEGIN
;  2256          3170  2
;  2257          3171  2 LOCAL
;  2258          3172  2         CLOSEST : REF $BBLOCK INITIAL (0),      ! Best checkpoint found
;  2259          3173  2         DSB     : REF $BBLOCK                   ! dynamic string block
;  2260          3174  2         ;
;  2261          3175  2
;  2262          3176  2
;  2263          3177  2 ! Initialize the queue pointer to the first item in the queue
;  2264          3178  2 !
;  2265          3179  2 DSB = .FLINK_ (SCB[PSM$Q_CHECKPOINT_QUEUE]);
;  2266          3180  2
;  2267          3181  2
;  2268          3182  2 ! Search the queue until we return to the queue header
;  2269          3183  2 !
;  2270          3184  2 UNTIL .DSB EQL SCB[PSM$Q_CHECKPOINT_QUEUE]
;  2271          3185  2 DO
;  2272          3186  3     BEGIN
;  2273          3187  3     BIND CKP = .DESC_ADDR_ (DSB[DSB_Q_DESC]) : $BBLOCK;
;  2274          3188  3
;  2275          3189  3     ! If this checkpoint preceeds the target page and is closer
;  2276          3190  3     ! than any other then save it
;  2277          3191  3     !
;  2278          3192  3     IF .CKP[SMBMSG$L_PAGE] LEQ .SCB[PSM$L_START_PAGE]
;  2279          3193  3     THEN
;  2280          3194  3         IF .CLOSEST EQL 0 THEN CLOSEST = CKP
;  2281          3195  3         ELSE
;  2282          3196  3             IF .CKP[SMBMSG$L_PAGE] GTRU .CLOSEST[SMBMSG$L_PAGE]
;  2283          3197  3             THEN
;  2284          3198  3                 CLOSEST = CKP;
;  2285          3199  3
;  2286          3200  3     ! Advance to the next queue entry
;  2287          3201  3     !
;  2288          3202  3     DSB = .FLINK_ (DSB[DSB_Q_QLINKS]);
```

```
;  2289          3203  2     END;
;  2290          3204  2
;  2291          3205  2
;  2292          3206  2     ! Return the address of the checkpoint if a useable one was found
;  2293          3207  2     !
;  2294          3208  2     IF .CLOSEST NEQ 0
;  2295          3209  2     THEN
;  2296          3210  2         ! If current page greater than target page,
;  2297          3211  2         ! or current page less than checkpoint page
;  2298          3212  2         !
;  2299          3213  2         IF .SCB[PSM$L_PAGE] GTRU .SCB[PSM$L_START_PAGE]
;  2300          3214  2         OR .SCB[PSM$L_PAGE] LSSU .CLOSEST[SMBMSG$L_PAGE]
;  2301          3215  2         THEN
;  2302          3216  2             .CLOSEST
;  2303          3217  2         ELSE
;  2304          3218  2             0
;  2305          3219  2     ELSE
;  2306          3220  2         0
;  2307          3221  2
;  2308          3222  1 END;


                                000C 00000 FIND_CHECKPOINT:
                                           .WORD   Save R2,R3                          ; 3166
                           50   D4 00002            CLRL    CLOSEST                     ; 3169
                     52    04   AC   D0 00004       MOVL    SCB, R2                     ; 3179
                     53  017C   C2   D0 00008       MOVL    380(R2), DSB
                     51  017C   C2   9E 0000D 1$:    MOVAB   380(R2), R1                 ; 3184
                     51          53   D1 00012       CMPL    DSB, R1
                                 1F   13 00015       BEQL    4$
                     51    0C   A3   D0 00017       MOVL    12(DSB), R1                 ; 3187
                   0224    C2   08   A1   D1 0001B   CMPL    8(R1), 548(R2)              ; 3192
                                      0E   14 00021  BGTR    3$
                                      50   D5 00023  TSTL    CLOSEST                     ; 3194
                                      07   13 00025  BEQL    2$
                     08    A0   08   A1   D1 00027   CMPL    8(R1), 8(CLOSEST)           ; 3196
                                      03   1B 0002C  BLEQU   3$
                           50         51   D0 0002E 2$:    MOVL    R1, CLOSEST         ; 3198
                           53         63   D0 00031 3$:    MOVL    (DSB), DSB          ; 3202
                                      D7   11 00034  BRB     1$                          ; 3184
                                      50   D5 00036 4$:    TSTL    CLOSEST             ; 3208
                                      11   13 00038  BEQL    5$
                   0224    C2  01EC   C2   D1 0003A   CMPL    492(R2), 548(R2)            ; 3213
                                      0A   1A 00041  BGTRU   6$
                     08    A0  01EC   C2   D1 00043   CMPL    492(R2), 8(CLOSEST)         ; 3214
                                      02   1F 00049  BLSSU   6$
                                      50   D4 0004B 5$:    CLRL    R0                  ; 3208
                                           04 0004D 6$:    RET                         ; 3222

; Routine Size:  78 bytes,    Routine Base:  CODE + 0D8B
```

```
;  2310          3223  1 %SBTTL 'GET_BUFFER - Get an output buffer (IOB)'
;  2311          3224  1 ! Functional Description:
;  2312          3225  1 !                 Allocates and initializes an IOB (Input/Output buffer
;  2313          3226  1 !                 control Block)
;  2314          3227  1 !
;  2315          3228  1 ! Formal Parameters:
;  2316          3229  1 !                 SCB     : SCB address
;  2317          3230  1 !
;  2318          3231  1 ! Implicit Inputs:
;  2319          3232  1 !                 none
;  2320          3233  1 !
;  2321          3234  1 ! Implicit Outputs:
;  2322          3235  1 !                 none
;  2323          3236  1 !
;  2324          3237  1 ! Returned Value:
;  2325          3238  1 !                 SS$_NORMAL if successful
;  2326          3239  1 !                 0 if no IOB's available
;  2327          3240  1 !
;  2328          3241  1 ! Side Effects:
;  2329          3242  1 !                 Allocates and initializes the IOB queue the first
;  2330          3243  1 !                 time this routine is called.
;  2331          3244  1 !--
;  2332          3245  1 ROUTINE GET_BUFFER (
;  2333          3246  1         SCB     : REF $BBLOCK
;  2334          3247  1         )       =
;  2335          3248  2 BEGIN
;  2336          3249  2
;  2337          3250  2 LOCAL
;  2338          3251  2         IOB     : REF $BBLOCK
;  2339          3252  2         ;
;  2340          3253  2
;  2341          3254  2 ! If their is already an IOB attached to the SCB then we are done
;  2342          3255  2 !
;  2343          3256  2 IF .SCB[PSM$A_IOB] NEQ 0
;  2344          3257  2 THEN
;  2345          3258  2     RETURN SS$_NORMAL;
;  2346          3259  2
;  2347          3260  2
;  2348          3261  2 ! If the queue has never been initialized then do it
;  2349          3262  2 !
;  2350          3263  2 IF .FLINK_ (SCB[PSM$Q_BUFFER_QUEUE]) EQL 0
;  2351          3264  2 THEN
;  2352          3265  3     BEGIN
;  2353          3266  3     INIT_QUEUE_HEADER_ (SCB[PSM$Q_BUFFER_QUEUE]);
;  2354          3267  3
;  2355          3268  3     ! Allocate as many IOB's for this SCB as specified by NUMOUTBUF
;  2356          3269  3     !
;  2357          3270  3     DECR I FROM PSM$K_NUMOUTBUF TO 1
;  2358          3271  3     DO
;  2359          3272  4         BEGIN
;  2360          3273  4         PSM$ALLOCATE_IOB (IOB, PSM$GL_MAXBUF);
;  2361          3274  4         IOB[IOB_A_CONTEXT] = .SCB;
;  2362          3275  4         INSERT_TAIL_ (IOB[IOB_Q_QLINKS], SCB[PSM$Q_BUFFER_QUEUE]);
;  2363          3276  3         END;
;  2364          3277  2     END;
;  2365          3278  2
;  2366          3279  2
```

```
2 ! Get an IOB, return if none available
2 !
2 IF REMOVE_HEAD_ (IOB, SCB[PSM$Q_BUFFER_QUEUE]) THEN RETURN 0;
2
2
2 ! Adjust the IOB address, clear the IOB flags, and attach the
2 ! IOB to the SCB.
2 !
2 IOB = .IOB - $BYTEOFFSET (IOB_Q_QLINKS);
2 IOB[IOB_L_FLAGS] = 0;
2 SCB[PSM$A_IOB] = .IOB;
2
2
2 ! Initialize the buffer descriptor
2 !
2 VECTOR [SCB[PSM$Q_OUTPUT_BUFFER], 0] = .DESC_SIZE_ (IOB[IOB_Q_BUFFER]);
2 VECTOR [SCB[PSM$Q_OUTPUT_BUFFER], 1] = .DESC_ADDR_ (IOB[IOB_Q_BUFFER]);
2
2 SS$_NORMAL
2
1 END;
```

```
                              0004 00000 GET_BUFFER:
                                                .WORD   Save R2                       ; 3245
          5E            04  C2 00002           SUBL2   #4, SP
          50       04   AC  D0 00005           MOVL    SCB, R0                       ; 3256
              01AC      C0  D5 00009           TSTL    428(R0)                       ;
                        64  12 0000D           BNEQ    4$                            ;
          50     0174   C0  9E 0000F           MOVAB   372(R0), R0                   ; 3263
                        60  D5 00014           TSTL    (R0)                          ;
                        2E  12 00016           BNEQ    2$                            ;
          60            50  D0 00018           MOVL    R0, (R0)                      ; 3266
     04   A0            50  D0 0001B           MOVL    R0, 4(R0)                     ;
          52            03  D0 0001F           MOVL    #3, I                         ; 3270
           00000000G    00  9F 00022 1$:       PUSHAB  PSM$GL_MAXBUF                 ; 3273
                  04    AE  9F 00028           PUSHAB  IOB                           ;
00000000G 00            02  FB 0002B           CALLS   #2, PSM$ALLOCATE_IOB          ;
          51            6E  D0 00032           MOVL    IOB, R1                       ; 3274
     14   A1       04   AC  D0 00035           MOVL    SCB, 20(R1)                   ;
          50       04   AC  D0 0003A           MOVL    SCB, R0                       ; 3275
   0178   D0            61  0E 0003E           INSQUE  (R1), @376(R0)                ;
          DC            52  F5 00043           SOBGTR  I, 1$                         ; 3270
          50       04   AC  D0 00046 2$:       MOVL    SCB, R0                       ; 3282
          6E     0174   D0  0F 0004A           REMQUE  @372(R0), IOB                 ;
                        03  1C 0004F           BVC     3$                            ;
                        50  D4 00051           CLRL    R0                            ;
                            04 00053           RET                                   ;
          51            6E  D0 00054 3$:       MOVL    IOB, R1                       ; 3289
                   2C   A1  D4 00057           CLRL    44(R1)                        ;
          50       04   AC  D0 0005A           MOVL    SCB, R0                       ; 3290
   01AC   C0            81  7E 0005E           MOVAQ   (R1)+, 428(R0)                ;
          50     01E0   C0  9E 00063           MOVAB   480(R0), R0                   ; 3295
          51            14  C0 00068           ADDL2   #20, R1                       ;
```

```
                                 60          61  3C 0006B         MOVZWL  (R1), (R0)                          ;
                              04 A0       04 A1  D0 0006E         MOVL    4(R1), 4(R0)                        ; 3296
                                 50          01  D0 00073 4$:     MOVL    #1, R0                              ; 3300
                                                 04 00076         RET                                         ;

; Routine Size:  119 bytes,    Routine Base:  CODE + 0DD9
```

```
1 %SBTTL 'HANDLER -- main signal handler'
1 ! Functional Description:
1 !             Catches signals, inhibits text expansion, and resignals
1 !
1 ! Formal Parameters:
1 !             STANDARD SIGNAL ARGUMENTS
1 !
1 ! Implicit Inputs:
1 !             none
1 !
1 ! Implicit Outputs:
1 !             none
1 !
1 ! Returned Value:
1 !             none
1 !
1 ! Side Effects:
1 !             none
1 !--
1 ROUTINE HANDLER (SIGARGS: REF BLOCK [, BYTE]) =
1
2 BEGIN
2
2 ! Disable expansion of error condition to text
2 !
2 SIGARGS [CHF$L_SIG_NAME] = .SIGARGS [CHF$L_SIG_NAME] OR STS$M_INHIB_MSG;
2
2 SS$_RESIGNAL
2
1 END;
```

```
                       0000 00000 HANDLER:.WORD   Save nothing          ; 3320
        50       04  AC  D0 00002         MOVL    SIGARGS, R0           ; 3326
    07  A0           10  88 00006         BISB2   #16, 7(R0)            ;
        50     0918  8F  3C 0000A         MOVZWL  #2328, R0             ; 3330
                         04 0000F         RET                           ;
```

; Routine Size: 16 bytes,   Routine Base: CODE + 0E50

```
1 %SBTTL 'PUTMSG_ACTION - action routine for $PUTMSG call'
1 ! Functional Description:
1 !             Adds carraige control and appends the messages into
1 !             the SCB.
1 !
1 ! Formal Parameters:
1 !             Standard $PUTMSG action routine interface
1 !
1 ! Implicit Inputs:
1 !             none
1 !
1 ! Implicit Outputs:
1 !             none
1 !
1 ! Returned Value:
1 !             none
1 !
1 ! Side Effects:
1 !             The message text is appended to the appropriate descriptor
1 !             in the SCB.
1 !--
1 ROUTINE PUTMSG_ACTION (
1         MSG_DESC        : REF $BBLOCK,
1         DYN_DESC
1         ) =
2 BEGIN
2
2 BIND FORMAT = $DESCRIPTOR ('!/!AS',  %CHAR (PSM$K_CHAR_CR));
2
2 LOCAL
2         WRK_DESC:       VECTOR [2],
2         WRK_BUFF:       VECTOR [512, BYTE]
2         ;
2
2 ! Setup a work descriptor
2 !
2 WRK_DESC [0] = %ALLOCATION (WRK_BUFF);
2 WRK_DESC [1] = WRK_BUFF;
2
2
2 ! Call FAO to add carriage control
2 !
2 $FAO (FORMAT, WRK_DESC, WRK_DESC, .MSG_DESC);
2
2
2 ! Append the resulting message to the specified descriptor
2 !
2 SIGNAL_IF_ERROR_ (STR$APPEND (.DYN_DESC, WRK_DESC));
2
2 RETURN 0;
2
1 END;
```

```
53  41  21  2F  21  00E60 P.AAS:   .ASCII  \!/!AS\
                 0D  00E65          .ASCII  <13>
```

```
                                    00E66                .BLKB   2
                                    00E68 P.AAR:         .LONG   6
                                    00E6C                .ADDRESS P.AAS

                                                 FORMAT=                 P.AAR
                                                         .EXTRN  SYS$FAO

                                    0004 00000 PUTMSG_ACTION:
                                                         .WORD   Save R2                         ; 3352
                         5E  FDF8   CE  9E 00002         MOVAB   -520(SP), SP
                      FB AD  0200   8F  3C 00007         MOVZWL  #512, WRK_DESC                  ; 3367
                      FC AD         6E  9E 0000D         MOVAB   WRK_BUFF, WRK_DESC+4            ; 3368
                            04      AC  DD 00011         PUSHL   MSG_DESC                        ; 3373
                            F8 AD   9F  DD 00014         PUSHAB  WRK_DESC
                            F8 AD   9F  DD 00017         PUSHAB  WRK_DESC
                            DB      AF  9F 0001A         PUSHAB  FORMAT
                00000000G   00      04  FB 0001D         CALLS   #4, SYS$FAO
                            F8 AD   9F  DD 00024         PUSHAB  WRK_DESC                        ; 3378
                            08      AC  DD 00027         PUSHL   DYN_DESC
                00000000G   00      02  FB 0002A         CALLS   #2, STR$APPEND
                            52      50  D0 00031         MOVL    R0, STATUS
                            09      52  E8 00034         BLBS    STATUS, 1$
                                    52  DD 00037         PUSHL   STATUS
                00000000G   00      01  FB 00039         CALLS   #1, LIB$SIGNAL
                                    50  D4 00040 1$:     CLRL    R0                              ; 3380
                                        04 00042         RET                                     ; 3382

; Routine Size:  67 bytes,    Routine Base:  CODE + 0E70
```

```
;  2473          3383  1 %SBTTL 'RESUME_SERVICE - Resume a previously suspended service'
;  2474          3384  1 ! Functional Description:
;  2475          3385  1 !               Resumes the input service at the top of the service
;  2476          3386  1 !               stack and resets the SCB values that were in effect
;  2477          3387  1 !               when the service was suspended.
;  2478          3388  1 !
;  2479          3389  1 ! Formal Parameters:
;  2480          3390  1 !               SCB     : SCB ADDRESS
;  2481          3391  1 !
;  2482          3392  1 ! Implicit Inputs:
;  2483          3393  1 !               Input service queue header
;  2484          3394  1 !
;  2485          3395  1 ! Implicit Outputs:
;  2486          3396  1 !               Context values that are preserved when a service is
;  2487          3397  1 !               suspended are restored.
;  2488          3398  1 !
;  2489          3399  1 ! Returned Value:
;  2490          3400  1 !               none
;  2491          3401  1 !
;  2492          3402  1 ! Side Effects:
;  2493          3403  1 !               The service is popped from the input service stack.
;  2494          3404  1 !--
;  2495          3405  1 ROUTINE RESUME_SERVICE (
;  2496          3406  1         SCB     : REF $BBLOCK
;  2497          3407  1         )       : NOVALUE =
;  2498          3408  2 BEGIN
;  2499          3409  2
;  2500          3410  2 LOCAL
;  2501          3411  2         DSB : REF $BBLOCK
;  2502          3412  2         ;
;  2503          3413  2
;  2504          3414  2 ! Decrement the depth and check for coding error
;  2505          3415  2 !
;  2506          3416  2 DECREMENT_ (SCB[PSM$B_INPUT_DEPTH]);
;  2507          3417  2 IF .SCB[PSM$B_INPUT_DEPTH] LSS 0
;  2508          3418  2 THEN
;  2509          3419  2     CODEERR_ ;
;  2510          3420  2
;  2511          3421  2
;  2512          3422  2 ! Release any dynamic memory of current stream
;  2513          3423  2 !
;  2514          3424  2 CLEAR_STRING_ (SCB[PSM$Q_INPUT_RECORD]);
;  2515          3425  2 CLEAR_STRING_ (SCB[PSM$Q_USER_RECORD]);
;  2516          3426  2
;  2517          3427  2
;  2518          3428  2 ! Get the context block for the previous stream
;  2519          3429  2 !
;  2520          3430  2 IF REMOVE_HEAD_ (DSB, SCB[PSM$Q_INPUT_QUEUE]) THEN CODEERR_ ;
;  2521          3431  2 DSB = .DSB - $BYTEOFFSET (DSB_Q_QLINKS);
;  2522          3432  2
;  2523          3433  2
;  2524          3434  2 ! Overlay the context area in the SCB
;  2525          3435  2 !
;  2526          3436  2 CH$MOVE (PSM$S_SERVICE_CONTEXT, .DESC_ADDR_ (DSB[DSB_Q_DESC]),
;  2527          3437  2     SCB[PSM$R_SERVICE_CONTEXT]);
;  2528          3438  2
;  2529          3439  2
```

```
;  2530          3440  2 ! Release the context block
;  2531          3441  2 !
;  2532          3442  2 PSM$DEALLOCATE_DSB (.DSB);
;  2533          3443  2
;  2534          3444  1 END;


                                01FC 00000 RESUME_SERVICE:
                                           .WORD   Save R2,R3,R4,R5,R6,R7,R8                    ; 3405
                 58 00000000G   00  9E 00002        MOVAB   LIB$STOP, R8
                 57 00000000G   00  9E 00009        MOVAB   STR$FREE1_DX, R7
                 52       04    AC  D0 00010        MOVL    SCB, R2                              ; 3416
                 50     02A5    C2  9E 00014        MOVAB   677(R2), R0
                                60  97 00019        DECB    (R0)
                                0B  18 0001B        BGEQ    1$                                   ; 3417
                                01  DD 0001D        PUSHL   #1                                   ; 3418
                    01061154    8F  DD 0001F        PUSHL   #17174868
                 68             02  FB 00025        CALLS   #2, LIB$STOP
                 01     0263    C2  91 00028 1$:    CMPB    611(R2), #1                          ; 3424
                                11  1A 0002D        BGTRU   2$
                 50     0260    C2  9E 0002F        MOVAB   608(R2), R0
                 60 020E0000    8F  D0 00034        MOVL    #34471936, (R0)
                          04    A0  D4 0003B        CLRL    4(R0)
                                10  11 0003E        BRB     3$
                 50             52  D0 00040 2$:    MOVL    R2, R0
                          0260  C0  B5 00043        TSTW    608(R0)
                                07  13 00047        BEQL    3$
                          0260  C2  9F 00049        PUSHAB  608(R2)
                 67             01  FB 0004D        CALLS   #1, STR$FREE1_DX
                 01     0273    C2  91 00050 3$:    CMPB    627(R2), #1                          ; 3425
                                11  1A 00055        BGTRU   4$
                 50     0270    C2  9E 00057        MOVAB   624(R2), R0
                 60 020E0000    8F  D0 0005C        MOVL    #34471936, (R0)
                          04    A0  D4 00063        CLRL    4(R0)
                                10  11 00066        BRB     5$
                 50             52  D0 00068 4$:    MOVL    R2, R0                               ; 1
                          0270  C0  B5 0006B        TSTW    624(R0)
                                07  13 0006F        BEQL    5$
                          0270  C2  9F 00071        PUSHAB  624(R2)
                 67             01  FB 00075        CALLS   #1, STR$FREE1_DX
                 56     0184    D2  0F 00078 5$:    REMQUE  @388(R2), DSB                        ; 3430
                                0B  1C 0007D        BVC     6$
                                01  DD 0007F        PUSHL   #1
                    01061154    8F  DD 00081        PUSHL   #17174868
                 68             02  FB 00087        CALLS   #2, LIB$STOP
                 50       04    AC  D0 0008A 6$:    MOVL    SCB, R0                              ; 3437
           0260  C0        0C  B6     1E  28 0008E  MOVC3   #30, @12(DSB), 608(R0)               ; 3442
                 00000000G  00        01  FB 00097  CALLS   #1, PSM$DEALLOCATE_DSB
                                      04  0009E     RET                                          ; 3444

; Routine Size:  159 bytes,    Routine Base:  CODE + 0EB3
```

```
;  2536          3445  1 %SBTTL 'SAVE_CHECKPOINT - Build a checkpoint item'
;  2537          3446  1 !++
;  2538          3447  1 ! Functional Description:
;  2539          3448  1 !               Builds a checkpoint item from values in the SCB and from
;  2540          3449  1 !               a READ_KEY operation to the current input service.
;  2541          3450  1 !
;  2542          3451  1 ! Formal Parameters:
;  2543          3452  1 !               SCB     : SCB address
;  2544          3453  1 !
;  2545          3454  1 ! Implicit Inputs:
;  2546          3455  1 !               none
;  2547          3456  1 !
;  2548          3457  1 ! Implicit Outputs:
;  2549          3458  1 !               none
;  2550          3459  1 !
;  2551          3460  1 ! Returned Value:
;  2552          3461  1 !               none
;  2553          3462  1 !
;  2554          3463  1 ! Side Effects:
;  2555          3464  1 !               none
;  2556          3465  1 !--
;  2557          3466  1 ROUTINE SAVE_CHECKPOINT (
;  2558          3467  1         SCB     : REF $BBLOCK
;  2559          3468  1         )       : NOVALUE =
;  2560          3469  2 BEGIN
;  2561          3470  2
;  2562          3471  2 LOCAL
;  2563          3472  2         CKP_DESC        : VECTOR [2],
;  2564          3473  2         KEY_DESC        : VECTOR [2] PRESET ([0]=0, [1]=0)
;  2565          3474  2         ;
;  2566          3475  2
;  2567          3476  2 BIND
;  2568          3477  2         IOB = .SCB[PSM$A_IOB] : $BBLOCK,                ! Current output blcok
;  2569          3478  2         CKP = IOB[IOB_T_CHECKPOINT_DATA] : $BBLOCK      ! Checkpoint area in IOB
;  2570          3479  2         ;
;  2571          3480  3 BEGIN
;  2572          3481  3
;  2573          3482  3 ! Locate the current input service
;  2574          3483  3 !
;  2575          3484  3 BIND SERVICE = PSM$SRV[.SCB[PSM$B_SERVICE_INDEX],0,0,0,0] : $BBLOCK;
;  2576          3485  3 LOCAL FUNCTION_STATUS;
;  2577          3486  3
;  2578          3487  3
;  2579          3488  3 ! Call the current input service to obtain the record key
;  2580          3489  3 !
;  2581          3490  3 FUNCTION_STATUS = BLISS (
;  2582          3491  3     .SERVICE[SRV_A_SERVICE],           !  - current input service
;  2583          3492  3     SCB,                               !  - SCB address by reference
;  2584          3493  3     SCB[PSM$R_USER_CONTEXT_AREA],      !  - user context area
;  2585          3494  3     UPLIT (PSM$K_GET_KEY),             !  - GET_KEY function
;  2586          3495  3     KEY_DESC,                          !  - output key desc
;  2587          3496  3     0);                                !  - <not used>
;  2588          3497  3
;  2589          3498  3 ! Case on the status
;  2590          3499  3 !
;  2591          3500  3 SELECTONEU .FUNCTION_STATUS OF
;  2592          3501  3     SET
```

```
;  2593          3502  3
;  2594          3503  3
;  2595          3504  3          ! Asynchronous read_key operations not allowed
;  2596          3505  3          !
;  2597          3506  3
;  2598          3507  3          [PSM$_PENDING]:
;  2599          3508  3              CODEERR_ ;
;  2600          3509  3
;  2601          3510  3
;  2602          3511  3          ! If not supported, then return that as our status
;  2603          3512  3          !
;  2604          3513  3
;  2605          3514  3          [PSM$_FUNNOTSUP]:
;  2606          3515  3              RETURN PSM$_FUNNOTSUP;
;  2607          3516  3
;  2608          3517  3
;  2609          3518  3          ! If errors then store them and return the error
;  2610          3519  3          !
;  2611          3520  3
;  2612          3521  3          [OTHERWISE]:
;  2613          3522  3              IF NOT .FUNCTION_STATUS
;  2614          3523  3              THEN
;  2615          3524  4                  BEGIN
;  2616          3525  4                  PSM$STORE_ERRORS (.SCB, .FUNCTION_STATUS);
;  2617          3526  4                  RETURN .FUNCTION_STATUS;
;  2618          3527  4                  END;
;  2619          3528  3          TES;
;  2620          3529  2      END;
;  2621          3530  2
;  2622          3531  2      ! We have a key -- check the size and copy it into
;  2623          3532  2      !
;  2624          3533  2      IF .KEY_DESC[0] GTRU SMBMSG$S_USER_KEY THEN CODEERR_ ;
;  2625          3534  2      CH$COPY (.KEY_DESC[0], .KEY_DESC[1], 0,
;  2626          3535  2          SMBMSG$S_USER_KEY, CKP[SMBMSG$Q_USER_KEY]);
;  2627          3536  2
;  2628          3537  2
;  2629          3538  2      ! Build the rest of the checkpoint
;  2630          3539  2      !
;  2631          3540  2      CKP[SMBMSG$B_CHECKPOINT_LEVEL] = SMBMSG$K_STRUCTURE_LEVEL;
;  2632          3541  2      CKP[SMBMSG$W_OFFSET] = .SCB_SIZE_ (USER_RECORD) - .SCB_SIZE_ (INPUT_RECORD);
;  2633          3542  2      CKP[SMBMSG$L_CARCON] = .SCB[PSM$L_CARCON];
;  2634          3543  2      CKP[SMBMSG$L_PAGE] = .SCB[PSM$L_PAGE];
;  2635          3544  2      CKP[SMBMSG$L_RECORD_NUMBER] = .SCB[PSM$L_RECORD_NUMBER];
;  2636          3545  2
;  2637          3546  2
;  2638          3547  2      ! Mark this IOB as having a checkpoint assoicated with it.
;  2639          3548  2      !
;  2640          3549  2      IOB[IOB_V_CHECKPOINT_PENDING] = 1;
;  2641          3550  2
;  2642          3551  2
;  2643          3552  2      ! Build a descriptor of the checkpoint
;  2644          3553  2      !
;  2645          3554  2      CKP_DESC[0] = SMBMSG$S_CHECKPOINT_DATA;
;  2646          3555  2      CKP_DESC[1] = CKP;
;  2647          3556  2
;  2648          3557  2
;  2649          3558  2      ! Place it in the checkpoint queue
```

```
;  2650          3559  2 !
;  2651          3560  2 ENQUEUE_CHECKPOINT (.SCB, CKP_DESC[0]);
;  2652          3561  2
;  2653          3562  2
;  2654          3563  2 SS$_NORMAL
;  2655          3564  2
;  2656          3565  1 END;


                                                00F52                P.AAT:  .BLKB   2
                                      00000006  00F54                        .LONG   6                                                       ;


                                                03FC 00000 SAVE_CHECKPOINT:
                                                                            .WORD   Save R2,R3,R4,R5,R6,R7,R8,R9                            ; 3465
                                                59       F7    AF  9E 00002         MOVAB   P.AAT, R9
                                                58 00000000G   00  9E 00006         MOVAB   LIB$STOP, R8
                                                5E             0C  C2 0000D         SUBL2   #12, SP
                                                               7E  D4 00010         CLRL    KEY_DESC                                        ; 3472
                                                         04    AE  D4 00012         CLRL    KEY_DESC+4
                                                50       04    AC  D0 00015         MOVL    SCB, R0                                         ; 3476
                                                57     01AC    C0  D0 00019         MOVL    428(R0), R7
                                                56       30    A7  9E 0001E         MOVAB   48(R7), R6                                      ; 3477
                                                51     027D    C0  9A 00022         MOVZBL  637(R0), R1                                     ; 3484
                                                51             10  C4 00027         MULL2   #16, R1
                                                   00000000G0041  9F  0002A         PUSHAB  PSM$SRV[R1]                                     ; 3491
                                                51             9E  D0 00031         MOVL    @(SP)+, R1
                                                               7E  D4 00034         CLRL    -(SP)                                           ; 3493
                                                         04    AE  9F 00036         PUSHAB  KEY_DESC
                                                               59  DD 00039         PUSHL   R9                                              ; 3494
                                                       02D0    C0  9F 0003B         PUSHAB  720(R0)                                         ; 3493
                                                         04    AC  9F 0003F         PUSHAB  SCB                                             ; 3490
                                                   61          05  FB 00042         CALLS   #5, (R1)                                        ; 3493
                                                   52          50  D0 00045         MOVL    R0, FUNCTION_STATUS
                                          00000000G  8F        52  D1 00048         CMPL    FUNCTION_STATUS, #PSM$_PENDING                  ; 3507
                                                               0D  12 0004F         BNEQ    1$
                                                               01  DD 00051         PUSHL   #1
                                                01061154       8F  DD 00053         PUSHL   #17174868
                                                   68          02  FB 00059         CALLS   #2, LIB$STOP
                                                               17  11 0005C         BRB     2$
                                          00000000G  8F        52  D1 0005E 1$:     CMPL    FUNCTION_STATUS, #PSM$_FUNNOTSUP                ; 3514
                                                               5F  13 00065         BEQL    4$
                                                   0B          52  E8 00067         BLBS    FUNCTION_STATUS, 2$                             ; 3522
                                                               52  DD 0006A         PUSHL   FUNCTION_STATUS                                 ; 3525
                                                         04    AC  DD 0006C         PUSHL   SCB
                                                FB1D       CF    02  FB 0006F         CALLS   #2, PSM$STORE_ERRORS
                                                               04  00074         RET                                                     ; 3526
                                                   08          6E  D1 00075 2$:     CMPL    KEY_DESC, #8                                    ; 3533
                                                               0B  1B 00078         BLEQU   3$
                                                               01  DD 0007A         PUSHL   #1
                                                01061154       8F  DD 0007C         PUSHL   #17174868
                                                   68          02  FB 00082         CALLS   #2, LIB$STOP
                                08       00       04    BE  6E  2C 00085 3$:     MOVC5   KEY_DESC, @KEY_DESC+4, #0, #8, 16(R6)           ; 3535
                                                         10  A6          00088
                                                   01    A6  01  90 0008D         MOVB    #1, 1(R6)                                       ; 3540
```

```
                          50       04  AC  D0 00091        MOVL    SCB, R0                              ; 3541
        02  A6      0270  C0     0260  C0  A3 00095        SUBW3   608(R0), 624(R0), 2(R6)              ;
                      04  A6     0278  C0  D0 0009E        MOVL    632(R0), 4(R6)                       ; 3542
                      08  A6     01EC  C0  D0 000A4        MOVL    492(R0), 8(R6)                       ; 3543
                      0C  A6     026C  C0  D0 000AA        MOVL    620(R0), 12(R6)                      ; 3544
                      2C  A7           01  88 000B0        BISB2   #1, 44(R7)                           ; 3549
                      08  AE           18  D0 000B4        MOVL    #24, CKP_DESC                        ; 3554
                      0C  AE           56  D0 000B8        MOVL    R6, CKP_DESC+4                       ; 3555
                                 08  AE  9F 000BC          PUSHAB  CKP_DESC                             ; 3560
                                     50  DD 000BF          PUSHL   R0                                   ;
                    FC7D  CF           02  FB 000C1        CALLS   #2, ENQUEUE_CHECKPOINT               ;
                                       04 000C6 4$:        RET                                          ; 3565

; Routine Size:  199 bytes,    Routine Base:  CODE + 0F58
```

```
;  2658          3566  1 %SBTTL 'SCHEDULE_SERVICE -- determine the next input service to process'
;  2659          3567  1 ! Functional Description:
;  2660          3568  1 !               Looks for an input service to process.  The primary list
;  2661          3569  1 !               of services is established by a bit vector.  Additional
;  2662          3570  1 !               sources of input are page headers, page setup, included modules,
;  2663          3571  1 !               and previously suspended input services.
;  2664          3572  1 !
;  2665          3573  1 ! Formal Parameters:
;  2666          3574  1 !               SCB     : SCB address
;  2667          3575  1 !
;  2668          3576  1 ! Implicit Inputs:
;  2669          3577  1 !               none
;  2670          3578  1 !
;  2671          3579  1 ! Implicit Outputs:
;  2672          3580  1 !               none
;  2673          3581  1 !
;  2674          3582  1 ! Returned Value:
;  2675          3583  1 !               SS$_NORMAL      - Service located
;  2676          3584  1 !               PSM$_EOF        - No input services remain
;  2677          3585  1 !
;  2678          3586  1 ! Side Effects:
;  2679          3587  1 !               An input service may be dequeued from the input stack,
;  2680          3588  1 !               or removed from the outstanding service list.
;  2681          3589  1 !--
;  2682          3590  1
;  2683          3591  1 ROUTINE SCHEDULE_SERVICE (
;  2684          3592  1         SCB     : REF $BBLOCK
;  2685          3593  1         )       =
;  2686          3594  2 BEGIN
;  2687          3595  2
;  2688          3596  2 BIND
;  2689          3597  2         LIST    = SCB[PSM$L_SERVICE_LIST] : BITVECTOR
;  2690          3598  2         ;
;  2691          3599  2
;  2692          3600  2 LOCAL
;  2693          3601  2         PIDX    : INITIAL (0)           ! Index into service list
;  2694          3602  2         ;
;  2695          3603  2
;  2696          3604  2 ! Reset values for new input service
;  2697          3605  2 !
;  2698          3606  2 SCB[PSM$L_RECORD_NUMBER] = 0;
;  2699          3607  2 SCB[PSM$V_READ_OFFSET] = 0;
;  2700          3608  2 SCB[PSM$V_FIRST_RECORD] = 1;
;  2701          3609  2 SCB[PSM$B_SERVICE_INDEX] = 0;
;  2702          3610  2
;  2703          3611  2
;  2704          3612  2 ! If there are any pending modules then select the LIBRARY_INPUT service
;  2705          3613  2 ! to process them.
;  2706          3614  2 !
;  2707          3615  2 IF STRIP_COMMA_DELIMITED_ITEM (SCB[PSM$Q_MODULE_LIST], SCB[PSM$Q_MODULE_NAME])
;  2708          3616  2 THEN
;  2709          3617  3     BEGIN
;  2710          3618  3     SCB[PSM$B_SERVICE_INDEX] = PSM$K_LIBRARY_INPUT;
;  2711          3619  3     RETURN SS$_NORMAL;
;  2712          3620  2     END;
;  2713          3621  2
;  2714          3622  2
```

```
;  2715          3623  2 ! If page setup has been requested then schedule it
;  2716          3624  2 !
;  2717          3625  2 IF TESTBITSC (LIST[PSM$K_PAGE_SETUP])
;  2718          3626  2 THEN
;  2719          3627  3     BEGIN
;  2720          3628  3     SCB[PSM$B_SERVICE_INDEX] = PSM$K_PAGE_SETUP;
;  2721          3629  3     RETURN SS$_NORMAL;
;  2722          3630  2     END;
;  2723          3631  2
;  2724          3632  2
;  2725          3633  2 ! Similarily, if page header has been requested then schedule it
;  2726          3634  2 !
;  2727          3635  2 IF TESTBITSC (LIST[PSM$K_PAGE_HEADER])
;  2728          3636  2 THEN
;  2729          3637  3     BEGIN
;  2730          3638  3     SCB[PSM$B_SERVICE_INDEX] = PSM$K_PAGE_HEADER;
;  2731          3639  3     RETURN SS$_NORMAL;
;  2732          3640  2     END;
;  2733          3641  2
;  2734          3642  2
;  2735          3643  2 ! If there is a suspended input service then resume it
;  2736          3644  2 !
;  2737          3645  2 IF .SCB[PSM$B_INPUT_DEPTH] GTRU 0
;  2738          3646  2 THEN
;  2739          3647  3     BEGIN
;  2740          3648  3     RESUME_SERVICE (.SCB);
;  2741          3649  3     RETURN SS$_NORMAL;
;  2742          3650  2     END;
;  2743          3651  2
;  2744          3652  2
;  2745          3653  2 ! This is a brand new input service -- reset values
;  2746          3654  2 !
;  2747          3655  2 SCB[PSM$L_PAGE] = 1;
;  2748          3656  2 SCB[PSM$L_PRINT_FLAGS] = 0;
;  2749          3657  2 SCB[PSM$L_L_MARGIN] = 0;
;  2750          3658  2 SCB[PSM$L_T_MARGIN] = 0;
;  2751          3659  2
;  2752          3660  2
;  2753          3661  2 ! Scan the service list for a pending input service
;  2754          3662  2 !
;  2755          3663  2 UNTIL FFS (PIDX, UPLIT (PSM$K_MAX), LIST, PIDX) ! False until list empty
;  2756          3664  2 DO
;  2757          3665  3     BEGIN
;  2758          3666  3     SCB[PSM$B_SERVICE_INDEX] = .PIDX;
;  2759          3667  3     LIST[.PIDX] = 0;
;  2760          3668  3     IF .PSM$SRV[.PIDX, SRV_ _SERVICE] NEQ 0
;  2761          3669  3     THEN
;  2762          3670  3         RETURN SS$_NORMAL;
;  2763          3671  2     END;
;  2764          3672  2
;  2765          3673  2
;  2766          3674  2 ! No service found, return EOF
;  2767          3675  2 !
;  2768          3676  2 PSM$_EOF
;  2769          3677  2
;  2770          3678  1 END;
```

```
                                                0101F                .BLKB   1
                                      00000017  01020 P.AAU:         .LONG   23                                                      ;


                                                003C 00000 SCHEDULE_SERVICE:
                                                           .WORD    Save R2,R3,R4,R5                                               ; 3591
                          53      04      AC    D0 00002           MOVL    SCB, R3                                                ; 3597
                          55    0218      C3    9E 00006           MOVAB   536(R3), R5                                            ;
                                          52    D4 0000B           CLRL    PIDX                                                   ;
                                0268      C3    D4 0000D           CLRL    620(R3)                                                ; 3606
                       11 A3              02    8A 00011           BICB2   #2, 17(R3)                                             ; 3607
                       10 A3              20    88 00015           BISB2   #32, 16(R3)                                            ; 3608
                          54    027D      C3    9E 00019           MOVAB   637(R3), R4                                            ; 3609
                                          64    94 0001E           CLRB    (R4)                                                   ;
                                01D4      C3    9F 00020           PUSHAB  468(R3)                                                ; 3615
                                01CC      C3    9F 00024           PUSHAB  460(R3)                                                ;
                    0000V CF              02    FB 00028           CALLS   #2, STRIP_COMMA_DELIMITED_ITEM                         ;
                          05              50    E9 0002D           BLBC    R0, 1$                                                 ;
                          64              03    90 00030           MOVB    #3, (R4)                                               ; 3618
                                          50    11 00033           BRB     7$                                                     ; 3619
                    05    65              01    E5 00035 1$:       BBCC    #1, (R5), 2$                                           ; 3625
                          64              01    90 00039           MOVB    #1, (R4)                                               ; 3628
                                          47    11 0003C           BRB     7$                                                     ; 3629
                    05    65              02    E5 0003E 2$:       BBCC    #2, (R5), 3$                                           ; 3635
                          64              02    90 00042           MOVB    #2, (R4)                                               ; 3638
                                          3E    11 00045           BRB     7$                                                     ; 3639
                                02A5      C3    95 00047 3$:       TSTB    677(R3)                                                ; 3645
                                          09    13 0004B           BEQL    4$                                                     ;
                                          53    DD 0004D           PUSHL   R3                                                     ; 3648
                          FE3B    CF              01    FB 0004F           CALLS   #1, RESUME_SERVICE                             ;
                                          2F    11 00054           BRB     7$                                                     ; 3649
                          01EC    C3              01    D0 00056 4$:       MOVL    #1, 492(R3)                                    ; 3655
                                0204      C3    D4 0005B           CLRL    516(R3)                                                ; 3656
                                01BC      C3    D4 0005F           CLRL    444(R3)                                                ; 3657
                                0230      C3    D4 00063           CLRL    560(R3)                                                ; 3658
                    52    65      91    AF              52    EA 00067 5$:       FFS     PIDX, P.AAU, (R5), PIDX                ; 3663
                                          1A    13 0006D           BEQL    8$                                                     ;
                          64              52    90 0006F           MOVB    PIDX, (R4)                                             ; 3666
                    00    65              52    E5 00072           BBCC    PIDX, (R5), 6$                                         ; 3667
                    50    52              04    78 00076 6$:       ASHL    #4, PIDX, R0                                           ; 3668
                                00000000G0040   9F 0007A           PUSHAB  PSM$SRV[R0]                                            ;
                                          9E    D5 00081           TSTL    @(SP)+                                                 ;
                                          E2    13 00083           BEQL    5$                                                     ;
                          50              01    D0 00085 7$:       MOVL    #1, R0                                                 ; 3670
                                          04    00088              RET                                                            ;
                          50 00000000G    8F    D0 00089 8$:       MOVL    #PSM$_EOF, R0                                          ; 3678
                                          04    00090              RET                                                            ;

; Routine Size:  145 bytes,    Routine Base:  CODE + 1024
```

```
;  2772          3679  1 %SBTTL 'SEARCH_FOR_STRING - Search for a string in a buffer'
;  2773          3680  1 ! Functional Description:
;  2774          3681  1 !                   This routine looks for a search string in the current
;  2775          3682  1 !                   input record.  It maintains context across calls so that
;  2776          3683  1 !                   strings that cross record boundaries can be located.
;  2777          3684  1 !
;  2778          3685  1 ! Formal Parameters:
;  2779          3686  1 !                   SCB     : SCB address
;  2780          3687  1 !                   KEY     : descriptor of search key
;  2781          3688  1 !                   TARGET  : descriptor of input record
;  2782          3689  1 !
;  2783          3690  1 ! Implicit Inputs:
;  2784          3691  1 !                   SCB[PSM$Q_SEARCH_CONTEXT]       - context from last call
;  2785          3692  1 !
;  2786          3693  1 ! Implicit Outputs:
;  2787          3694  1 !                   none
;  2788          3695  1 !
;  2789          3696  1 ! Returned Value:
;  2790          3697  1 !                   SS$_NORMAL      - the KEY was found in the TARGET
;  2791          3698  1 !                   0               - KEY was not found
;  2792          3699  1 ! Side Effects:
;  2793          3700  1 !                   none
;  2794          3701  1 !--
;  2795          3702  1 GLOBAL ROUTINE SEARCH_FOR_STRING (
;  2796          3703  1           SCB     : REF $BBLOCK,
;  2797          3704  1           KEY     : REF $BBLOCK,
;  2798          3705  1           TARGET  : REF $BBLOCK
;  2799          3706  1           ) =
;  2800          3707  2 BEGIN
;  2801          3708  2
;  2802          3709  2 LOCAL   PTR;
;  2803          3710  2
;  2804          3711  2 ! Append the input record to the context from the last call
;  2805          3712  2 !
;  2806          3713  2 STR$APPEND (SCB[PSM$Q_SEARCH_CONTEXT], .TARGET);
;  2807          3714  2
;  2808          3715  2
;  2809          3716  2 ! Compress white space (blanks and tabs) to a single space and upcase
;  2810          3717  2 !
;  2811          3718  2 BAS$EDIT (SCB[PSM$Q_SEARCH_CONTEXT], SCB[PSM$Q_SEARCH_CONTEXT], EDIT_MASK);
;  2812          3719  2
;  2813          3720  2
;  2814          3721  2 ! Look for the key as a substring of the target
;  2815          3722  2 !
;  2816          3723  2 PTR = CH$FIND_SUB (
;  2817          3724  2           .SCB_SIZE_ (SEARCH_CONTEXT),    ! Target appended to remainder
;  2818          3725  2           .SCB_ADDR_ (SEARCH_CONTEXT),
;  2819          3726  2           .DESC_SIZE_ (.KEY),             ! Search key
;  2820          3727  2           .DESC_ADDR_ (.KEY)
;  2821          3728  2                   );
;  2822          3729  2
;  2823          3730  2 ! Extract the last few characters of the input record as the context
;  2824          3731  2 ! for the next call
;  2825          3732  2 !
;  2826          3733  2 STR$RIGHT (
;  2827          3734  2           SCB[PSM$Q_SEARCH_CONTEXT],
;  2828          3735  2           SCB[PSM$Q_SEARCH_CONTEXT],
```

```
;  2829     3736  2          %REF (.SCB_SIZE_ (SEARCH_CONTEXT) - .DESC_SIZE_ (.KEY) + 1)
;  2830     3737  2          );
;  2831     3738  2
;  2832     3739  2
;  2833     3740  2      ! Return 0 if not found, SS$_normal if located
;  2834     3741  2      !
;  2835     3742  2      IF CH$FAIL (.PTR)
;  2836     3743  2      THEN
;  2837     3744  2          0
;  2838     3745  2      ELSE
;  2839     3746  2          SS$_NORMAL
;  2840     3747  2
;  2841     3748  1      END;


                                                  007C 00000     .ENTRY  SEARCH_FOR_STRING, Save R2,R3,R4,R5,R6      ; 3702
                                              5E  04   C2 00002  SUBL2   #4, SP
                                                  AC   DD 00005  PUSHL   TARGET                                      ; 3713
                                           50 04  AC   D0 00008  MOVL    SCB, R0
                                           55 0210 C0   9E 0000C  MOVAB   528(R0), R5
                                                  55   DD 00011  PUSHL   R5
                                 00000000G 00     02   FB 00013  CALLS   #2, STR$APPEND
                                                  30   DD 0001A  PUSHL   #48                                         ; 3718
                                                  55   DD 0001C  PUSHL   R5
                                                  55   DD 0001E  PUSHL   R5
                                 00000000G 00     03   FB 00020  CALLS   #3, BASSEDIT
                                           54 08  AC   D0 00027  MOVL    KEY, R4                                    ; 3726
                                           56     64   3C 0002B  MOVZWL  (R4), R6
                                 04 B5     65 04  B4   56   39 0002E  MATCHC  R6, @4(R4), (R5), @4(R5)               ; 3727
                                                  03   13 00035  BEQL    1$
                                           53     56   D0 00037  MOVL    R6, R3
                                           53     56   C2 0003A  1$:     SUBL2   R6, R3
                                           50     65   3C 0003D  MOVZWL  (R5), R0                                   ; 3736
                                           51     64   3C 00040  MOVZWL  (R4), R1
                                           50     51   C2 00043  SUBL2   R1, R0
                                           6E 01  A0   9E 00046  MOVAB   1(R0), (SP)
                                              4020 8F   BB 0004A  PUSHR   #^M<R5,SP>                                 ; 3735
                                                  55   DD 0004E  PUSHL   R5
                                 00000000G 00     03   FB 00050  CALLS   #3, STR$RIGHT
                                                  53   D5 00057  TSTL    PTR                                        ; 3742
                                                  03   12 00059  BNEQ    2$
                                                  50   D4 0005B  CLRL    R0
                                                       04 0005D  RET
                                           50     01   D0 0005E  2$:     MOVL    #1, R0
                                                       04 00061  RET                                                 ; 3748

; Routine Size:  98 bytes,    Routine Base:  CODE + 10B5
```

```
;  2843     3749  1 %SBTTL 'STRIP_COMMA_DELIMITED_ITEM -- remove item from comma separate list'
;  2844     3750  1 ! Functional Description:
;  2845     3751  1 !               This routine removes one item from the front of a comma
;  2846     3752  1 !               separated list.
;  2847     3753  1 !
;  2848     3754  1 ! Formal Parameters:
;  2849     3755  1 !               INPUT   : descriptor of input list
;  2850     3756  1 !               OUTPUT  : removed item
;  2851     3757  1 !
;  2852     3758  1 ! Implicit Inputs:
;  2853     3759  1 !               none
;  2854     3760  1 !
;  2855     3761  1 ! Implicit Outputs:
;  2856     3762  1 !               The INPUT list is rewritten with the item removed
;  2857     3763  1 !
;  2858     3764  1 ! Returned Value:
;  2859     3765  1 !               none
;  2860     3766  1 !
;  2861     3767  1 ! Side Effects:
;  2862     3768  1 !               none
;  2863     3769  1 !--
;  2864     3770  1 ROUTINE STRIP_COMMA_DELIMITED_ITEM (
;  2865     3771  1         INPUT   : REF $BBLOCK,
;  2866     3772  1         OUTPUT  : REF $BBLOCK
;  2867     3773  1         ) =
;  2868     3774  2 BEGIN
;  2869     3775  2
;  2870     3776  2 LOCAL PTR;
;  2871     3777  2
;  2872     3778  2 ! If nothing to do then return
;  2873     3779  2 !
;  2874     3780  2 IF .DESC_SIZE_ (.INPUT) EQL 0 THEN RETURN 0;
;  2875     3781  2
;  2876     3782  2
;  2877     3783  2 ! Locate the first comma or end of string
;  2878     3784  2 !
;  2879     3785  2 PTR = CH$FIND_CH (.DESC_SIZE_ (.INPUT), .DESC_ADDR_ (.INPUT), %C ',');
;  2880     3786  2
;  2881     3787  2
;  2882     3788  2 ! If no comma found the the entire input string is the resultant item
;  2883     3789  2 ! and the input descriptor can be released
;  2884     3790  2 !
;  2885     3791  2 IF CH$FAIL (.PTR)
;  2886     3792  2 THEN
;  2887     3793  3     BEGIN
;  2888     3794  3     COPY_DX_DX_ (.INPUT, .OUTPUT);
;  2889     3795  3     STR$FREE1_DX (.INPUT);
;  2890     3796  3     END
;  2891     3797  2 ELSE
;  2892     3798  2     ! Comma found -- move the item from input list to output list
;  2893     3799  2     !
;  2894     3800  3     BEGIN
;  2895     3801  3     PTR = .PTR - .DESC_ADDR_ (.INPUT);
;  2896     3802  3     STR$LEFT (.OUTPUT, .INPUT, PTR);
;  2897     3803  3     PTR = .PTR + 2;
;  2898     3804  3     STR$RIGHT (.INPUT, .INPUT, PTR);
;  2899     3805  2     END;
```

```
; 2900          3806  2
; 2901          3807  2
; 2902          3808  2  ! Return success
; 2903          3809  2  !
; 2904          3810  2  SS$_NORMAL
; 2905          3811  2
; 2906          3812  1  END;



                                          000C 00000 STRIP_COMMA_DELIMITED_ITEM:
                                                               .WORD   Save R2,R3                         ; 3770
                              5E          04  C2 00002         SUBL2   #4, SP                             ;
                              52      04  AC  D0 00005         MOVL    INPUT, R2                          ; 3780
                                          62  B5 00009         TSTW    (R2)                               ;
                                          03  12 0000B         BNEQ    1$                                 ;
                                          50  D4 0000D         CLRL    R0                                 ;
                                              04 0000F         RET                                        ;
              04   B2         62          2C  3A 00010 1$:     LOCC    #44, (R2), @4(R2)                  ; 3785
                                          02  12 00015         BNEQ    2$                                 ;
                                          51  D4 00017         CLRL    R1                                 ;
                              6E          51  D0 00019 2$:     MOVL    R1, PTR                            ;
                                          26  12 0001C         BNEQ    4$                                 ; 3791
                                          52  DD 0001E         PUSHL   R2                                 ; 3794
                                      08  AC  DD 00020         PUSHL   OUTPUT                             ;
                   00000000G  00          02  FB 00023         CALLS   #2, STR$COPY_DX                    ;
                              53          50  D0 0002A         MOVL    R0, STATUS                         ;
                              09          53  E8 0002D         BLBS    STATUS, 3$                         ;
                                          53  DD 00030         PUSHL   STATUS                             ;
                   00000000G  00          01  FB 00032         CALLS   #1, LIB$SIGNAL                     ;
                                          52  DD 00039 3$:     PUSHL   R2                                 ; 3795
                   00000000G  00          01  FB 0003B         CALLS   #1, STR$FREE1_DX                   ;
                                          22  11 00042         BRB     5$                                 ; 3791
                              6E      04  A2  C2 00044 4$:     SUBL2   4(R2), PTR                         ; 3801
                                    4004  8F  BB 00048         PUSHR   #^M<R2,SP>                         ; 3802
                                      08  AC  DD 0004C         PUSHL   OUTPUT                             ;
                   00000000G  00          03  FB 0004F         CALLS   #3, STR$LEFT                       ;
                              6E          02  C0 00056         ADDL2   #2, PTR                            ; 3803
                                    4004  8F  BB 00059         PUSHR   #^M<R2,SP>                         ; 3804
                                          52  DD 0005D         PUSHL   R2                                 ;
                   00000000G  00          03  FB 0005F         CALLS   #3, STR$RIGHT                      ;
                              50          01  D0 00066 5$:     MOVL    #1, R0                             ; 3812
                                              04 00069         RET                                        ;

; Routine Size:  106 bytes,    Routine Base:  CODE + 1117
```

```
;  2908          3813  1 %SBTTL 'SUSPEND_SERVICE -- suspend the current input service'
;  2909          3814  1 ! Functional Description:
;  2910          3815  1 !                   Suspends the current input service by placing its
;  2911          3816  1 !                   context on an input service stack.
;  2912          3817  1 !
;  2913          3818  1 ! Formal Parameters:
;  2914          3819  1 !                   SCB     : SCB address
;  2915          3820  1 !
;  2916          3821  1 ! Implicit Inputs:
;  2917          3822  1 !                   none
;  2918          3823  1 !
;  2919          3824  1 ! Implicit Outputs:
;  2920          3825  1 !                   none
;  2921          3826  1 !
;  2922          3827  1 ! Returned Value:
;  2923          3828  1 !                   none
;  2924          3829  1 !
;  2925          3830  1 ! Side Effects:
;  2926          3831  1 !                   The current service is placed on the stack
;  2927          3832  1 !--
;  2928          3833  1 GLOBAL ROUTINE SUSPEND_SERVICE (
;  2929          3834  1         SCB     : REF $BBLOCK
;  2930          3835  1         ) : NOVALUE =
;  2931          3836  2 BEGIN
;  2932          3837  2
;  2933          3838  2 LOCAL
;  2934          3839  2         DSB : REF $BBLOCK
;  2935          3840  2         ;
;  2936          3841  2
;  2937          3842  2
;  2938          3843  2 ! Increment the stack depth and check for overflow
;  2939          3844  2 !
;  2940          3845  2 INCREMENT_ (SCB[PSM$B_INPUT_DEPTH]);
;  2941          3846  2 IF .SCB[PSM$B_INPUT_DEPTH] GTR 15
;  2942          3847  2 THEN
;  2943          3848  3     BEGIN
;  2944          3849  3     PSM$STORE_ERRORS (.SCB, PSM$_TOOMANYLEV, 1, .SCB[PSM$L_RECORD_NUMBER]);
;  2945          3850  3     RETURN;
;  2946          3851  2     END;
;  2947          3852  2
;  2948          3853  2
;  2949          3854  2 ! Get a Dynamic String control Block and copy the service context area into it.
;  2950          3855  2 !
;  2951          3856  2 PSM$ALLOCATE_DSB (DSB);
;  2952        P 3857  2 COPY_R_DX_ (OPLIT WORD (PSM$S_SERVICE_CONTEXT), SCB[PSM$R_SERVICE_CONTEXT],
;  2953          3858  2     DSB[DSB_Q_DESC]);
;  2954          3859  2
;  2955          3860  2
;  2956          3861  2 ! Place it in the input queue
;  2957          3862  2 !
;  2958          3863  2 INSERT_HEAD_ (DSB[DSB_Q_QLINKS], SCB[PSM$Q_INPUT_QUEUE]);
;  2959          3864  2
;  2960          3865  2
;  2961          3866  2 ! Clear the service context area
;  2962          3867  2 !
;  2963          3868  2 CH$FILL (0, PSM$S_SERVICE_CONTEXT, SCB[PSM$R_SERVICE_CONTEXT]);
;  2964          3869  2
```

```
; 2965          3870  1 END;


                                              01181           .BLKB   1
                                       001E   01182 P.AAV:    .WORD   30                                              ;


                                       003C   00000           .ENTRY  SUSPEND_SERVICE, Save R2,R3,R4,R5               ; 3833
              5E                   04  C2     00002           SUBL2   #4, SP                                          ;
              52            04     AC  D0     00005           MOVL    SCB, R2                                         ; 3845
              50          02A5     C2  9E     00009           MOVAB   677(R2), R0                                     ;
                                   60  96     0000E           INCB    (R0)                                            ;
              0F                   60  91     00010           CMPB    (R0), #15                                       ; 3846
                                   14  15     00013           BLEQ    1$                                              ;
                          026C     C2  DD     00015           PUSHL   620(R2)                                         ; 3849
                                   01  DD     00019           PUSHL   #1                                              ;
                     00000000G     8F  DD     0001B           PUSHL   #PSM$_TOOMANYLEV                                ;
                                   52  DD     00021           PUSHL   R2                                              ;
         F93D   CF                 04  FB     00023           CALLS   #4, PSM$STORE_ERRORS                            ;
                                       04     00028           RET                                                     ; 3848
                                   5E  DD     00029 1$:       PUSHL   SP                                              ; 3856
      00000000G 00                 01  FB     0002B           CALLS   #1, PSM$ALLOCATE_DSB                            ;
                          0260     C2  9F     00032           PUSHAB  608(R2)                                         ; 3858
                            C5     AF  9F     00036           PUSHAB  P.AAV                                           ;
           7E       08      AE     08  C1     00039           ADDL3   #8, DSB, -(SP)                                  ;
      00000000G 00                 03  FB     0003E           CALLS   #3, STR$COPY_R                                  ;
                53                 50  D0     00045           MOVL    R0, STATUS                                      ;
                09                 53  E8     00048           BLBS    STATUS, 2$                                      ;
                                   53  DD     0004B           PUSHL   STATUS                                          ;
      00000000G 00                 01  FB     0004D           CALLS   #1, LIB$SIGNAL                                  ;
           0184 D2          00     BE  0E     00054 2$:       INSQUE  @DSB, @388(R2)                                  ; 3863
                50          04     AC  D0     0005A           MOVL    SCB, R0                                         ; 3868
   1E       00  6E                 00  2C     0005E           MOVC5   #0, (SP), #0, #30, 608(R0)                      ;
                          0260     C0         00063                                                                   ;
                                       04     00066           RET                                                     ; 3870

; Routine Size:  103 bytes,    Routine Base:  CODE + 1184
```

```
; 2967          3871  1 END
; 2968          3872  0 ELUDOM


                                                        .EXTRN  LIB$SIGNAL, LIB$STOP

;                               PSECT SUMMARY
;
;       Name                    Bytes                   Attributes
;
;  CODE                           4587  NOVEC,NOWRT,  RD ,  EXE,NOSHR,  LCL,  REL,  CON,NOPIC,ALIGN(2)


;                             Library Statistics
;                                          -------- Symbols --------      Pages       Processing
;       File                               Total    Loaded   Percent      Mapped      Time
;
;  _$255$DUA28:[SYSLIB]LIB.L32;1            18619       113         0        1000        00:01.9



;                                       COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS$:DISPATCH/OBJ=OBJ$:DISPATCH MSRC$:DISPATCH/UPDATE=(ENH$:DISPATCH)

; Size:         4342 code + 245 data bytes
; Run Time:         01:40.9
; Elapsed Time:     04:30.3
; Lines/CPU Min:     2302
; Lexemes/CPU-Min: 24505
; Memory Used:  746 pages
; Compilation Complete
```

BANNER
LIS

PLIWRITE
LIS

SMBREQ
REQ

PLIVECTOR
LIS

SMBSRVSHR
MAP

FORMAT
LIS

PLIRODATA
LIS

SMBDEF
SDL

PRTSMB

DISPATCH
LIS

PLISTRING
LIS

PRTSMB
MAP